n8n

လိုတိုရှင်း

အိမောင်

FAIRWAY

ပုံနှိပ်မှတ်တမ်း

ထုတ်ဝေသူ	-	ဦးအိမောင် Fairway စာပေ (၀၃၃၀၇) အမှတ် 3A308၊ ရတနာနှင်းဆီအိမ်ရာ၊ ရတနာလမ်း၊ ဒဂုံဆိပ်ကမ်းမြို့နယ်၊ ရန်ကုန်တိုင်းဒေသကြီး။
ပုံနှိပ်သူ	-	ဦးအိမောင် Ebook (ဒစ်ဂျစ်တယ်စာအုပ်)
မျက်နှာဖုံးဒီဇိုင်း	-	အိမောင် (Fairway)
ကွန်ပျူတာစာစီ	-	အိမောင် (Fairway)
တန်ဖိုး	-	၃၀၀၀၀ ကျပ်
ဖြန့်ချီရေး	-	Fairway စာပေ (၀၉ - ၂၅၂ ၄၂၆ ၃၈၈)

မိတ်ဆက်
2

အခန်း (၁) - Docker နှင့် Install ပြုလုပ်ခြင်း
8

အခန်း (၂) - NPM နှင့် Install ပြုလုပ်ခြင်း
27

အခန်း (၃) - JSON
31

အခန်း (၄) - API
34

အခန်း (၅) - First Workflow
42

အခန်း (၆) - Weather Chat
59

အခန်း (၇) - Invoice & Reminder Workflow
75

အခန်း (၈) - Support Ticket Workflow
104

အခန်း (၉) - AI Agent Workflow
124

အခန်း (၁၀) - AI နှင့် Workflow များဖန်တီးခြင်း
148

အခန်း (၁၁) - ngrok နှင့် အသုံးပြုခြင်း
157

နိဂုံးချုပ်
171

မိတ်ဆက်

n8n ဆိုတာ Workflow Automation နည်းပညာ ဖြစ်ပါတယ်။ လူကိုယ်တိုင် အဆင့်ဆင့် လုပ်ရတဲ့အလုပ်တွေကို Manual လုပ်စရာမလိုတော့ဘဲ အလိုအလျောက် လုပ်သွားအောင် စီစဉ်ထားလို့ရတဲ့ နည်းပညာပါ။

ဥပမာ - ကိုယ့်ဆီက တစ်ခုခု ဝယ်ယူလိုသူ Customer က ဆက်သွယ်လာတဲ့အခါ၊

၁။ သင့်တော်သလို အကြောင်းပြန်၊

၂။ အချက်အလက်တွေကို မှတ်ထား၊

၃။ နောက်ရက်တွေမှာ Follow-Up လိုက်ပြီး ပြန်ဆက်သွယ်၊

၄။ ဝယ်ယူဖြစ်ခဲ့ရင် မှတ်ထား၊

၅။ မဝယ်ယူဖြစ်ခဲ့ရင်လည်း မှတ်ထား၊

....စတဲ့ အလုပ်တွေကို လူကိုယ်တိုင် တစ်ခုချင်း၊ တစ်ဆင့်ချင်း လုပ်နေရတယ်ဆိုရင်၊ ဒီအဆင့်တွေ အားလုံးကို အလိုအလျောက် လုပ်သွားအောင် n8n နဲ့ စီစဉ်ထားလို့ရနိုင်ပါတယ်။

Customer က ဆက်သွယ်လာတာနဲ့ ကိုယ်တစ်ချက်မှ ဝင်ကိုင်စရာမလိုဘဲ Workflow တွေက အလိုအလျောက် အသက်ဝင်ပြီး၊ အကြောင်းပြန်ပေးတာ၊ မှတ်သားတာ၊ နောက်ရက်တွေမှာ Follow-Up လိုက်တာမျိုးတွေ အကုန်လုံးကို အလိုအလျောက် လုပ်ပေးနိုင်တာပါ။

ဒီအတွက် AI Agent နည်းပညာတွေ ပူးတွဲအသုံးပြုလို့ရသလို၊ AI Agent နည်းပညာတွေမလိုဘဲနဲ့ လုပ်လို့ရနိုင်တာတွေလည်း အများကြီးရှိနေပါတယ်။ ဒီစာအုပ်မှာ အဲ့ဒီလို အသုံးပြုနည်းတွေကို ဖော်ပြပေးသွားမှာပါ။

No Code

တကယ်တော့ Workflow Automation ဆိုတာ အသစ်အဆန်းတော့ မဟုတ်ပါဘူး။ ကျွန်တော်တို့လို Coding ပိုင်းကျွမ်းကျင်တဲ့ Developer တွေဆိုရင် ဆော့ဖ်ဝဲပရောဂျက်တွေမှာ လုပ်စရာရှိတဲ့ အလုပ်တွေကို Script တွေရေးပြီး အလိုအလျှောက် လုပ်ခိုင်းတာဟာ လုပ်နေကြ ထုံးစံ ဖြစ်ပါတယ်။

n8n ရဲ့ ထူးခြားချက်ကတော့ ကုဒ်တွေရေးစရာမလိုဘဲ Automation လုပ်ဆောင်ချက်ကို ရရှိနိုင်ခြင်းပဲ ဖြစ်ပါတယ်။ အလားတူနည်းပညာတွေ အများကြီးရှိကြပေမဲ့ လက်ရှိ

လူစိတ်ဝင်စားမှု အများဆုံးနည်းပညာဖြစ်သလို နိုင်ငံတကာအဆင့် လုပ်ငန်းကြီးတွေမှာ အသုံးပြုနေတဲ့ နည်းပညာလည်း ဖြစ်ပါတယ်။

History

n8n ကို အခုနောက်ပိုင်းမှာ လူသိပိုများလာပေမဲ့ (၂၀၁၉) ခုနှစ်လောက်ကတည်းက ထွက်ပေါ်ခဲ့တဲ့ နည်းပညာဖြစ်ပါ။ ChatGPT နောက်ပိုင်း AI နည်းပညာတွေ ခေတ်စားလာတော့မှ ရိုးရိုး Workflow တွေသာမက AI Agent တွေနဲ့ပါ ချိတ်ဆက် အလုပ်လုပ်နိုင်သွားလို့ ရုတ်တရက်ပိုပြီး လူစိတ်ဝင်စားမှု ပိုများသွားခဲ့တာ ဖြစ်ပါတယ်။

n8n ဆိုတဲ့နာမည်က ထူးဆန်းနေရင်၊ အဲ့ဒါ Nodemation လို့ခေါ်တဲ့ အမည်ရဲ့ အတိုကောက်လို့ မှတ်နိုင်ပါတယ်။ ရှေ့က n နဲ့ စ၊ နောက်က n နဲ့ ဆုံးပြီး အလယ်မှာ စာလုံး (၈) လုံး ပါလို့ n8n လို့ခေါ်တာပါ။

နည်းပညာဘက်မှာ အလားတူ အမည်ပေးနည်းတွေကို သုံးကြလေ့ရှိပါတယ်။ ဥပမာ -

- **l10n** - Localization
- **i18n** - Internationalization

Use Cases

n8n ကို သုံးလို့ရနိုင်တဲ့ နေရာတွေ အများကြီး ရှိပါတယ်။ လုပ်ငန်းတွေအတွက်ဆိုရင် Sales, Marketing, Operation, Accounting, Customer Support, Supply Chain Management စသည်ဖြင့် ကဏ္ဍစုံမှာ အသုံးချနိုင်ပါတယ်။ လုပ်ငန်းပိုင်းတွေ လက်ရှိ

ကျွမ်းကျင်တဲ့ စာဖတ်သူတွေဆိုရင် ဒီအပိုင်းတွေမှာ ကျွန်တော်စာရေးသူထက်တောင် ပိုထိရောက်အောင် အသုံးချနိုင်ဦးမှာပါ။

တစ်ကိုယ်ရေအသုံးပြုမှု အနေနဲ့ဆိုရင်လည်း သတင်းအချက်အလက်တွေ အလိုအလျောက် စုဆောင်းဖို့၊ စိတ်ကူးအိုင်ဒီယာတွေ အလိုအလျောက် Generate လုပ်ဖို့၊ Social Media Content တွေ အလိုအလျောက်ဖန်တီးဖို့ စသည့်ဖြင့် အသုံးချလို့ရနိုင်ပါတယ်။

Software Developer တွေဆိုရင်လည်း ကိုယ်ဖန်တီးထားတဲ့ ဆော့ဖ်ဝဲ API တွေကို Third-Party API တွေနဲ့ အလိုအလျောက် ချိတ်ဆက်လုပ်ဆောင်တဲ့ စနစ်တွေ ဖန်တီးလို့ရနိုင်ပါတယ်။ ဒါကြောင့် ကဏ္ဍစုံက လူပေါင်းစုံအတွက် အသုံးဝင်မယ့် နည်းပညာတစ်ခုပဲ ဖြစ်ပါတယ်။

Fair Use

n8n ဟာ Fair Use ခေါ် အခကြေးငွေ ပေးစရာမလိုဘဲ ရယူအသုံးပြုခွင့် ပေးထားတဲ့ နည်းပညာတစ်ခုပါ။ Open Source ဆိုတာကို ကြားဖူးကြပါလိမ့်မယ်။ ဆော့ဖ်ဝဲတစ်ခုရဲ့ Source Code တွေကိုပါ ရယူခွင့်ပေးထားတဲ့ ဆော့ဖ်ဝဲအမျိုးအစားတွေပါ။

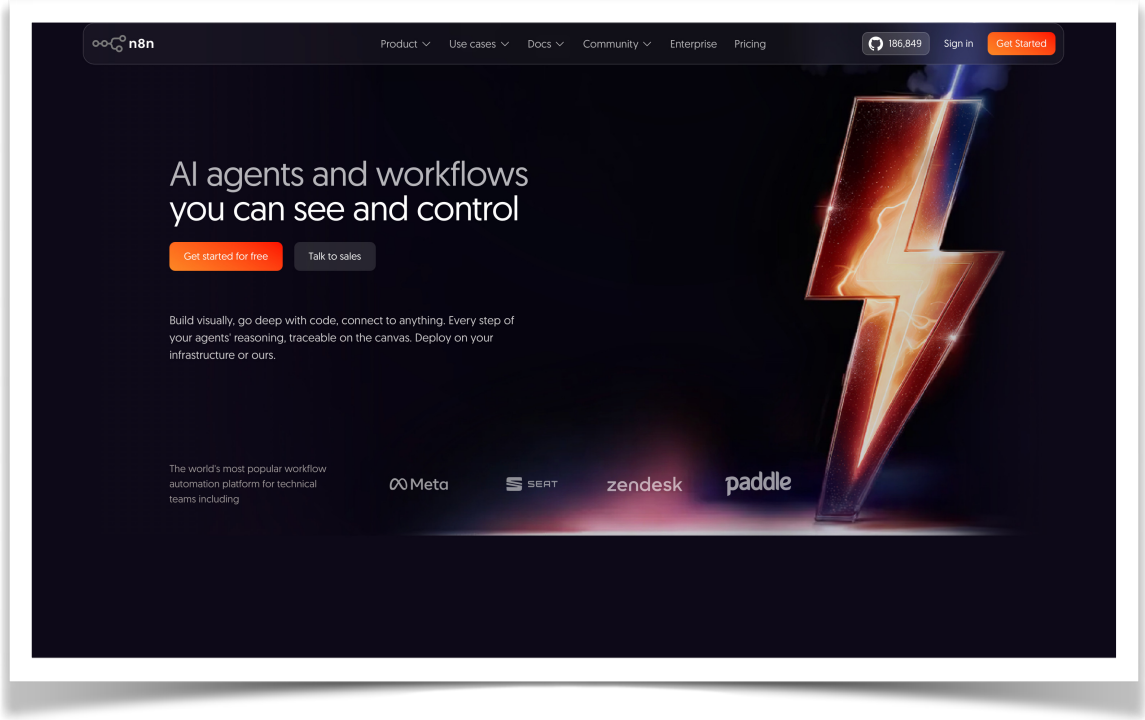
n8n က Open Source စစ်စစ်တော့ မဟုတ်ပါဘူး။ သူကုဒ်ကို ယူသုံးခွင့်ပေးထားပါတယ်။ သူနဲ့စင်ပြိုင် လုပ်ငန်းထူထောင်ခွင့်ကိုတော့ မပေးဘဲ ကန့်သတ်ထားပါတယ်။ ဒါကြောင့် Open Source လို့မခေါ်ဘဲ Fair Use နည်းပညာလို့ ခေါ်ကြတာပါ။

ကျွန်တော်တို့က n8n နဲ့ စင်ပြိုင်လုပ်ငန်း ထူထောင်ဖို့ အသုံးပြုမှာ မဟုတ်ဘဲ၊ သူ့နည်းပညာကို ကိုယ့်လိုအပ်ချက်အတွက် အသုံးပြုမှာဖြစ်တဲ့အတွက် အသုံးပြုခွင့်ရှိပါတယ်။

Local n8n

ဒီစာအုပ်မှာ n8n ကို ကိုယ့်စက်ထဲမှာ Install လုပ်ပြီး အသုံးပြုတဲ့နည်းကို အဓိကထား ဖော်ပြသွားမှာပါ။ ကွန်ပျူတာ မရှိလို့ဘဲဖြစ်ဖြစ်၊ အစပိုင်း Install မလုပ်ချင်သေးလို့ဘဲဖြစ်ဖြစ်၊ စမ်းကြည့်ချင်ရင် အောက်ပါလင့်ကနေတစ်ဆင့် သူ့ရဲ့ Platform မှာ အကောင့်ဆောက်ပြီး အသုံးပြုနိုင်ပါတယ်။

https://n8n.io/



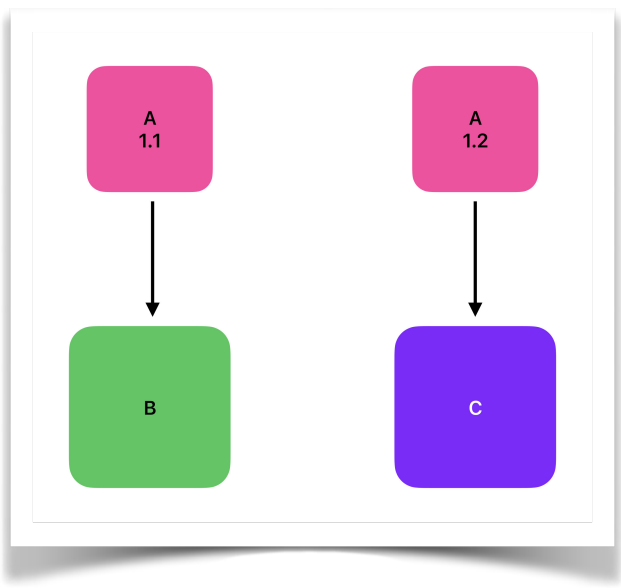
အကောင့်တည်ဆောက်ပြီးတာနဲ့ နှစ်ပတ်စမ်းသပ်အသုံးပြုခွင့် ပေးပါတယ်။ နှစ်ပတ် ကျော်ရင်တော့ သတ်မှတ်လစဉ်ကြေးနဲ့ဆက်လက်အသုံးပြုရပါတယ်။

အကြံပေးချင်တာကတော့၊ အခမဲ့ရတဲ့ နှစ်ပတ်နဲ့ သူ့ပလက်ဖောင်းမှာပဲ အရင်စမ်းကြည့်ပြီး နောက်ပိုင်းကျတော့မှ ကိုယ့်စက်ထဲမှာ Install လုပ်ပြီး ဆက်လက်အသုံးပြုလိုက်ပါ။

အခန်း (၁) - Docker နှင့် Install ပြုလုပ်ခြင်း

n8n ကို နည်းလမ်းနှစ်မျိုးနဲ့ ကိုယ့်စက်ထဲမှာ Install လုပ်လို့ရတဲ့အတွက် နှစ်မျိုးလုံးကို ထည့်သွင်းဖော်ပြပါမယ်။ ပထမတစ်နည်းကတော့ Docker လို့ခေါ်တဲ့ နည်းပညာကို အသုံးပြုတဲ့ နည်းပါ။

Docker ဆိုတာ Container နည်းပညာလို့ ခေါ်ပါတယ်။ လိုတိုရှင်း ဒီလို မှတ်နိုင်ပါတယ်။ ဆော့ဖ်ဝဲ A, B နဲ့ C ဆိုပြီး သုံးခုရှိတယ်လို့ စိတ်ကူးကြည့်ပါ။

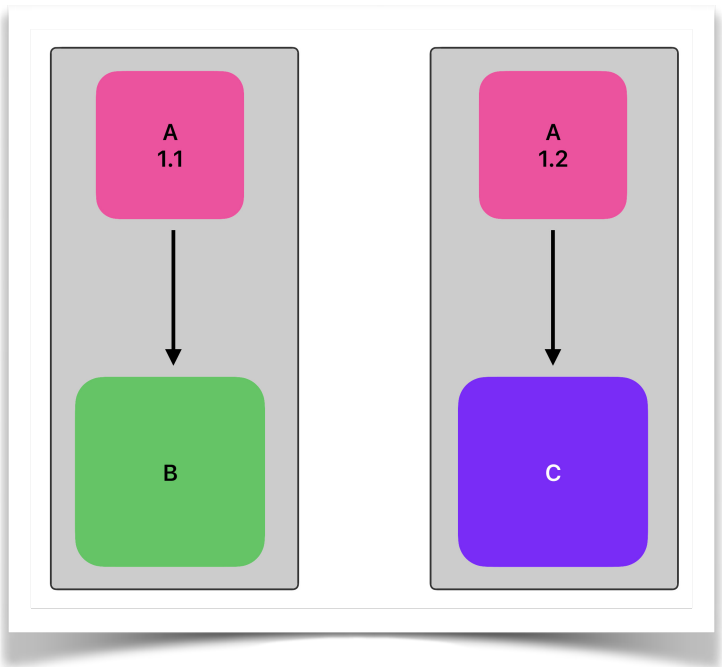


နမူနာပုံကို ကြည့်လိုက်ပါ။ ဆော့ဖ်ဝဲ B ရော C က ပါ ဆော့ဖ်ဝဲ A ကို အသုံးပြုထားကြပါတယ်။ ပြဿနာက အသုံးပြုထားတဲ့ Version မတူကြတာပါ။

ဒီတော့ ကိုယ့်စက်ထဲမှာ A 1.1 ကို ထည့်ထားရင် B ကို သုံးလို့ရမယ်။ A 1.2 ကို ထည့်ထားရင် C ကို သုံးလို့ရမယ်။ 1.1 နဲ့ 1.2 နှစ်မျိုးလုံးတော့ အမျိုးအစားတူနေလို့ အတူတူထည့်လို့မရဘူး ဆိုတာ ဖြစ်နိုင်ပါတယ်။

Docker က အဲ့ဒီပြဿနာကို ရှင်းပေးပါတယ်။

Docker က Software တွေကို တိုက်ရိုက်မ Run ဘဲ Container လို့ခေါ်တဲ့ Box လေးတစ်ခုထဲမှာ ထည့်ပြီးတော့ Run ပေးတယ် လို့စိတ်ကူးကြည့်နိုင်ပါတယ်။ အဲ့ဒီ Box လေးထဲမှာ လိုအပ်တာတွေအကုန် Install လုပ်ထားလို့ ရပါတယ်။



ဒါကြောင့် A 1.1 ကို ကိုယ့်စက်ထဲမှာ တိုက်ရိုက် Install မလုပ်ဘဲ B နဲ့အတူ သူ့ Box ထဲမှာ Install လုပ်လို့ရသွားသလို၊ A 1.2 ကိုလည်း C နဲ့အတူ သူ့ Box ထဲမှာပဲ Install လုပ်လို့ရသွားမှာ ဖြစ်ပါတယ်။

ဒီလိုသဘောတရားတွေကြောင့် Docker ဟာ အရေးပါတဲ့ နည်းပညာတစ်ခုပါ။

Docker ဟာ Linux နည်းပညာဖြစ်ပြီး Windows တို့ Mac တို့မှာ အသုံးပြုလိုရင် **Docker Desktop** လို့ခေါ်တဲ့ ထပ်ဆင့်နည်းပညာတစ်ခုကို အသုံးပြုရပါတယ်။ ပုံမှန်အားဖြင့် Docker Desktop ကို Download ရယူပြီး Install လုပ်လိုက်ရင် အသုံးပြုဖို့ Ready ဖြစ်သွားပါပြီ။

ဒါပေမဲ့ လက်တွေ့မှာ ထင်သလောက် ရိုးရှင်းလွယ်ကူမှု မရှိပါဘူး။ Docker က Virtualization လို့ ခေါ်တဲ့ နည်းပညာတစ်မျိုးကို အသုံးပြုထားပြီး Windows အသုံးပြုသူတွေအတွက် ဒါနဲ့ပတ်သက်ပြီး ဖြစ်နိုင်ခြေ ပြဿနာတချို့ ရှိနေပါတယ်။

- စက်အဟောင်းလေးနဲ့ သုံးနေတာမို့လို့ ကိုယ့်ကွန်ပျူတာရဲ့ CPU က Virtualization လုပ်ဆောင်ချက် မပါတာ၊
- စက်ရဲ့ BIOS က Update မဖြစ်လို့ Virtualization Support မရတာ၊ ဒါမှမဟုတ် စက်ရဲ့ BIOS ထဲမှာ Virtualization ကို ပိတ်ထားမိတာ၊
- Windows Version က နိမ့်နေပြီး Update မဖြစ်လို့ Docker က Support မလုပ်တာ၊

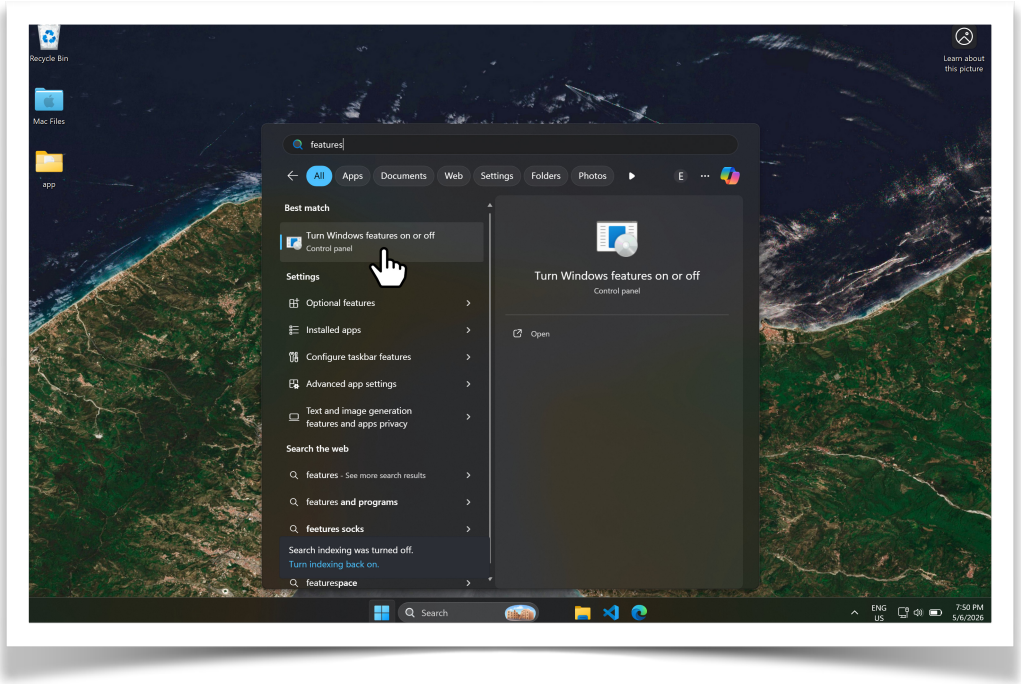
- Windows Install လုပ်စဉ် အကြောင်းအမျိုးမျိုးကြောင့် Component တွေ Library တွေ စုံအောင်ပါမလာတာ၊

...စသည်ဖြင့် တစ်ခုခု အဆင်မပြေဖြစ်ခဲ့ရင် ဘာကြောင့်လဲဆိုတာ အစကောက်အဖြေရှာရ ခက်လေ့ ရှိပါတယ်။

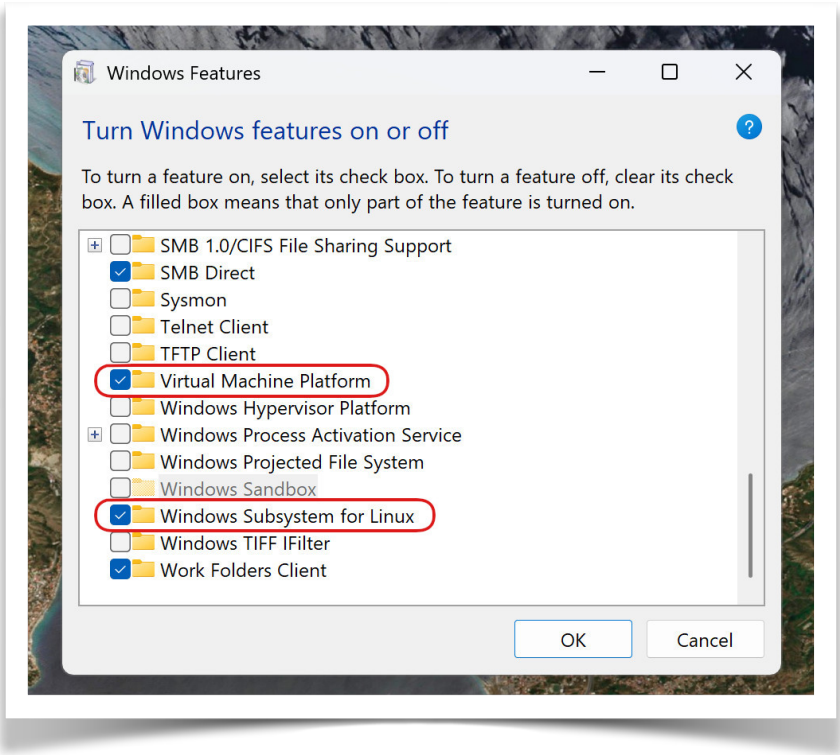
ကွန်ပျူတာကလည်း သင့်တင့်တဲ့ ကွန်ပျူတာဖြစ်မယ်။ Windows ကလည်း Update ဖြစ်မယ်ဆိုရင်တော့ Docker Desktop ကို Download ရယူပြီး ကိုယ့် Windows ကွန်ပျူတာမှာ အခက်အခဲ သိပ်မရှိဘဲ အသုံးပြုလို့ရမှာပါ။

Install ပြုလုပ်ဖို့အတွက် လိုအပ်တဲ့ အဆင့်တွေကို ဖော်ပြပေးပါမယ်။

အဆင့် (၁)



ပထမဆုံးအနေနဲ့ Windows Start Menu ကနေ **Turn Windows features on or off** ကို ရှာပြီး ဖွင့်လိုက်ပါ။ ပြီးတဲ့အခါ Virtual Machine Platform နဲ့ Windows Subsystem for Linux တို့က ပုံမှာ ပြထားသလို Check ဖြစ်နေရပါမယ်။



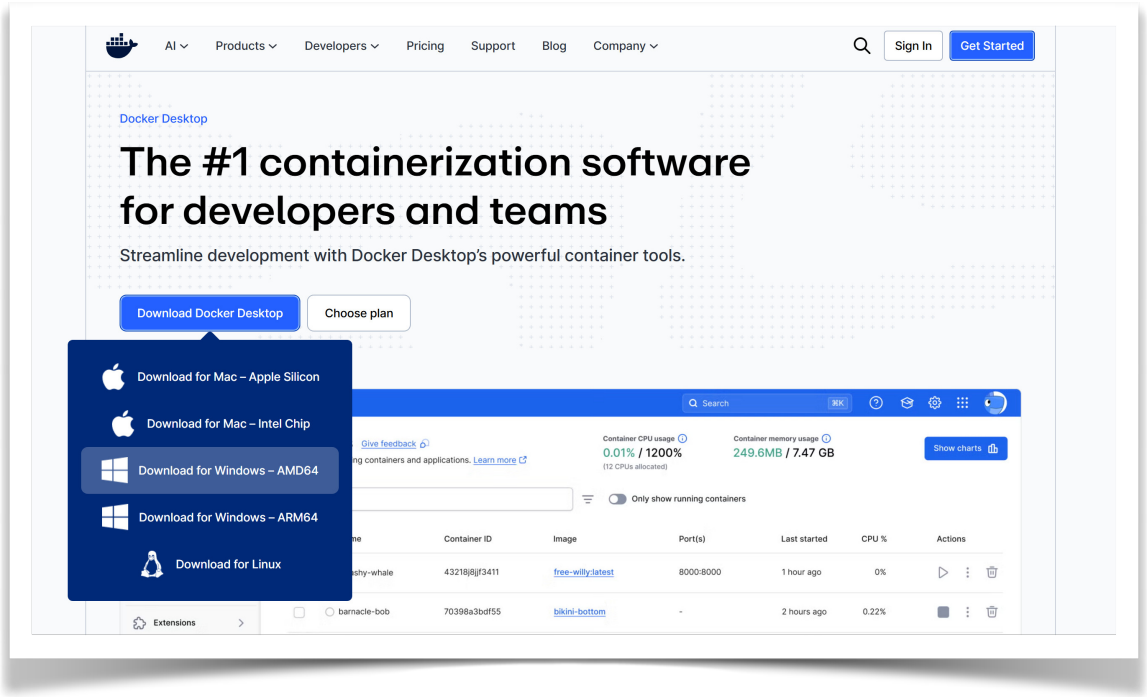
OK နှိပ်လိုက်ရင် Windows က ဒီလုပ်ဆောင်ချက်တွေ အသက်ဝင်ဖို့ လိုအပ်တာတွေ Install လုပ်သွားပါလိမ့်မယ်။ ပြီးတဲ့အခါ စက်ကို Restart လုပ်ပေးဖို့ လိုပါတယ်။

Docker ဟာ အထက်မှာပြောခဲ့သလို Linux နည်းပညာဖြစ်ပါတယ်။ Windows မှာ Run လို့ရဖို့အတွက် WSL ခေါ် Linux ကို Windows ထဲမှာ ထပ်ဆင့်သုံးလို့ရတဲ့ နည်းပညာတွေ လိုအပ်ပါတယ်။ ဒါကြောင့် အဲ့ဒီနည်းပညာတွေ မဖွင့်ရသေးရင် ဖွင့်ပေးရတဲ့သဘောပါ။

အဆင့် (၂)

ဆက်လက်ပြီး Docker Desktop ကို အောက်ပါလင့်ကနေ ဒေါင်းလိုက်ပါ။

https://www.docker.com/products/docker-desktop/



Download လုပ်တဲ့အခါ Windows အတွက် AMD64 နဲ့ ARM64 ဆိုပြီး နှစ်မျိုး ရှိနိုင်ပါတယ်။ Windows ကွန်ပျူတာ အများစုက AMD64 တွေဖြစ်ပါလိမ့်မယ်။ သေချာအောင် ကိုယ့်ကွန်ပျူတာက ဘာအမျိုးအစားလည်း အရင်စစ်သင့်ပါတယ်။ စစ်ဆေးပြီးရင် သင့်တော်တဲ့ Installer ဖိုင်ကို ဒေါင်းလိုက်ပါ။

ပြီးတဲ့အခါ Install လုပ်ကြည့်ပါ။ ဒီအတိုင်း Install လုပ်လို့မရရင် Installer ဖိုင်ပေါ်မှာ Right-Click နှိပ်ပြီး **Run as Administrator** ကို ရွေးပြီး စမ်းကြည့်ပါ။

အဆင့် (၃)

စာရေးသူကိုယ်တိုင် စမ်းကြည့်တဲ့အခါ နောက်ဆုံး Version ကို Install လုပ်လို့မရတဲ့ ပြဿနာတစ်ခု ရှိနေပါတယ်။ အဲ့ဒီပြဿနာကို အခုလို ဖြေရှင်းလိုက်ရပါတယ်။

Command Prompt (သို့မဟုတ်) Terminal App ပေါ်မှာ Right-Click နှိပ်ပြီး **Run as Administrator** နဲ့ ဖွင့်ရပါတယ်။ ပြီးတဲ့အခါ ဒီ Command တွေကို Run ရပါတယ်။

```
rmdir C:\ProgramData\DockerDesktop
```

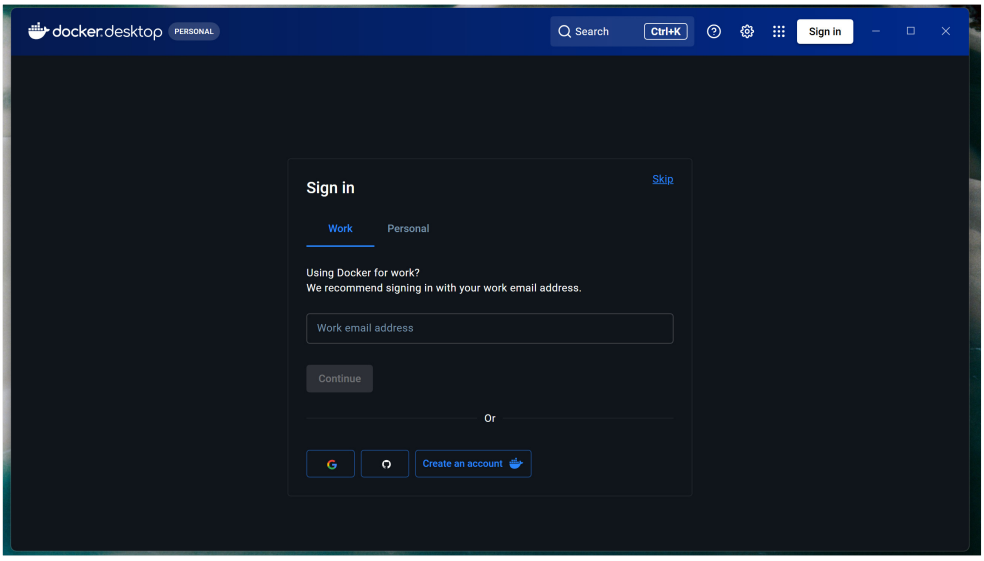
```
mkdir C:\ProgramData\DockerDesktop
```

ပထမ Command က စက်ထဲမှာ DockerDesktop အမည်နဲ့ ဖိုဒါရှိပြီး ဖြစ်နေမှာစိုးလို့ အရင်ဖျက်လိုက်တာပါ။ နောက် Command က DockerDesktop ဖိုဒါကို Admin User အနေနဲ့ ဖန်တီးလိုက်တာပါ။ ဒီတော့မှသာ Install လုပ်လို့ ရတာကို တွေ့ရပါတယ်။

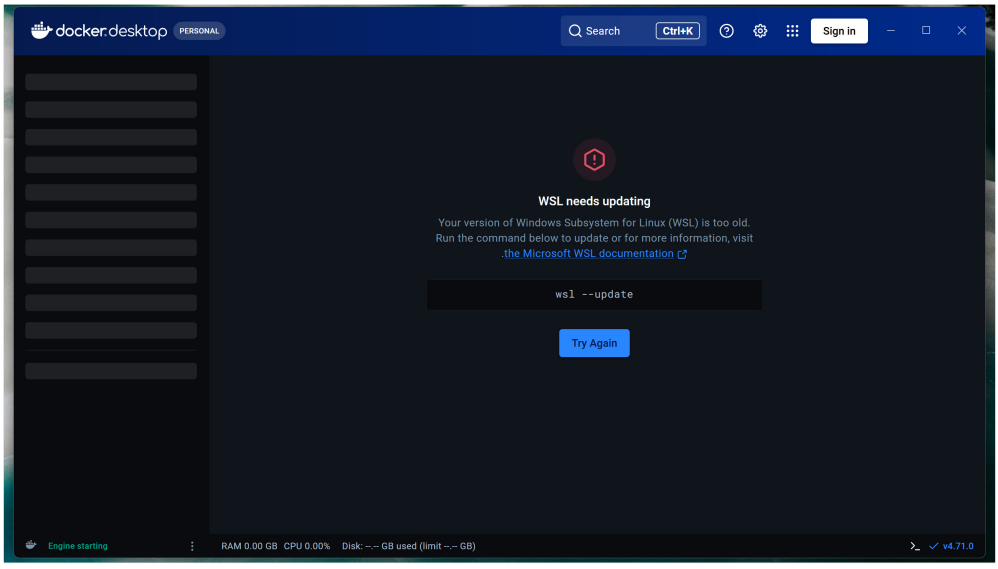
အဆင့် (၄)

Install လုပ်တဲ့ အဆင့်တွေကိုတော့ တစ်ဆင့်ချင်း မပြောတော့ပါဘူး။ ကိုယ်တိုင်ပဲ စမ်းကြည့်လိုက်ပါ။ Install လုပ်ပြီးသွားတဲ့အခါ စက်ကို Logout လုပ်ပြီး ပြန်ဝင်ပေးရပါတယ်။ Restart လုပ်လိုက်ရင်လည်း ရပါတယ်။

Install အောင်မြင်သွားလို့ Desktop ပေါ်က Docker Icon သို့မဟုတ် Start Menu ထဲက Docker Desktop ကို နှိပ်ပြီး ဖွင့်လိုက်တဲ့အခါ အခုလိုရလဒ်ကို တွေ့မြင်ရနိုင်ပါတယ်။

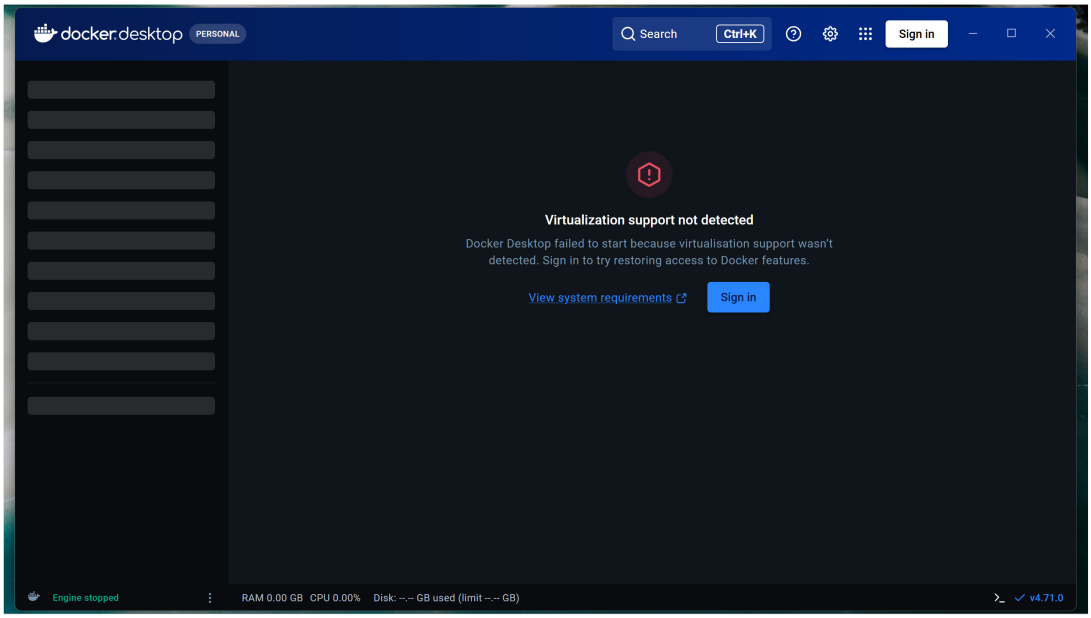


အကောင့်ဝင်ခိုင်းနေတာပါ။ မဝင်ချင်ရင် **Skip** ကို နှိပ်ပြီး ကျော်လိုက်လို့ရပါတယ်။ နောက်တစ်ဆင့်မှာ ဒီလိုရလဒ်မျိုးကို လာပြတာလည်း ဖြစ်နိုင်ပါတယ်။



ဒါက WSL ကို Update လုပ်ပေးပါလို့ ပြောနေတာပါ။ Run ရမယ့် Command လည်း တစ်ခါတည်း တွဲပြထားပါတယ်။ Command Prompt (သို့) Terminal App ကို Right-Click နှိပ်ပြီး **Run as Administrator** နဲ့ ဖွင့်လိုက်ပါ။ ပြီးရင် သူပေးထားတဲ့ Command ကို ကူးထည့်ပြီး Run လိုက်ပါ။ Windows က WSL ကို Update လုပ်သွားပါလိမ့်မယ်။

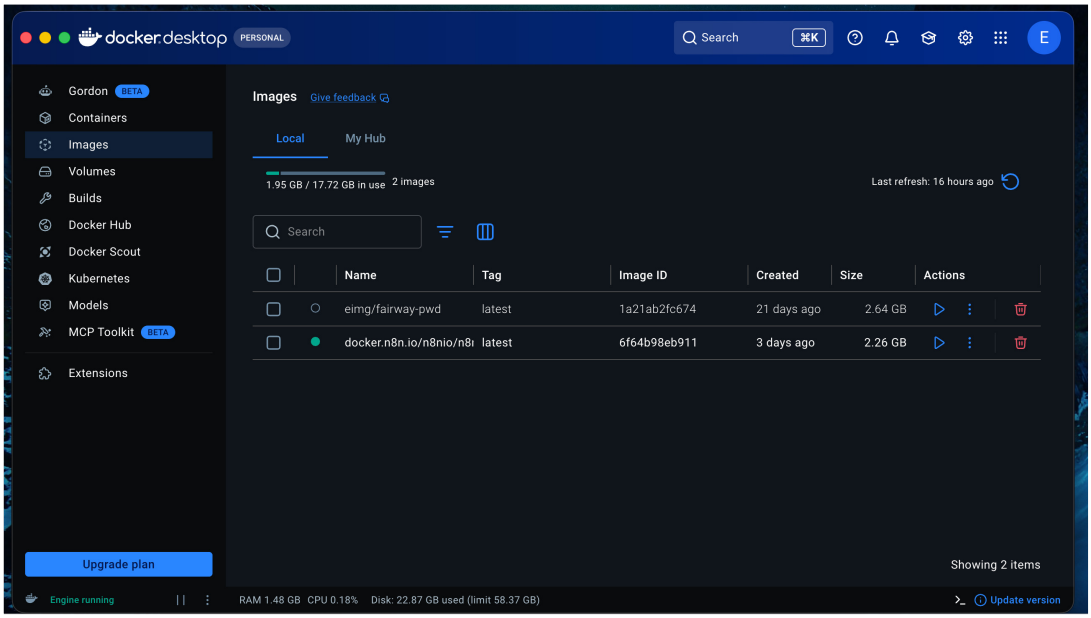
အကယ်၍များ နောက်တစ်ဆင့်မှာ အောက်ကပုံမှာ ပြထားသလို Virtualization Not Supported လို့များ ပြနေခဲ့ရင်တော့ အဆင်မပြေတာပါ။



ဘယ်လိုဖြေရှင်းရမလဲဆိုတဲ့ ဖြေရှင်းနည်းကို တိတိကျကျ ပြောလို့မရနိုင်တော့ဘဲ အထက်မှာ ပြောခဲ့တဲ့ ဖြစ်နိုင်ခြေအကြောင်းတွေ အများကြီးထဲက တစ်ခုခုကြောင့် ဖြစ်နိုင်သွားပါတယ်။

သိပ်စိတ်ညစ်မခံပါနဲ့။ Docker အဆင်မပြေလည်း n8n ကို တခြားနည်းနဲ့ Run လို့ရပါတယ်။ ဒါကြောင့် အဆင်မပြေရင် နောက်တစ်ခန်းကို ဆက်ကြည့်လိုက်ပါ။

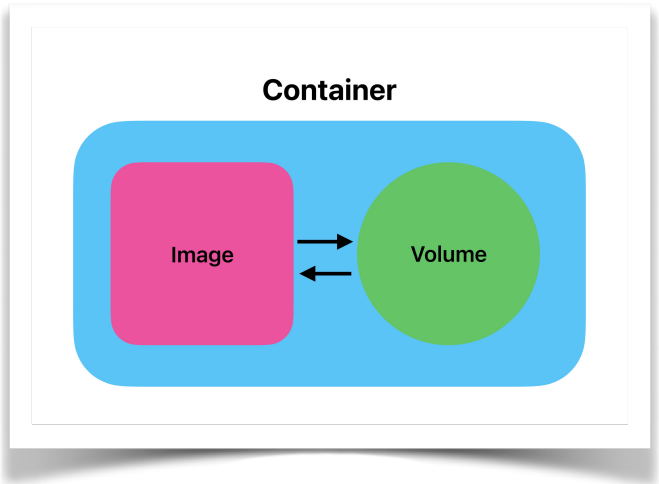
အားလုံးအဆင်ပြေတယ်ဆိုရင်တော့ အခုလိုရလဒ်မျိုးကို ရရှိမှာ ဖြစ်ပါတယ်။



စာရေးသူလက်ရှိသုံးလက်စကို နမူနာပြလိုက်တာဖြစ်လို့ အဲ့ဒီအတိုင်း အတိအကျတူစရာ မလိုပါဘူး။ နမူနာပုံမှာ ဘယ်ဘက်ခြမ်း Menu ကနေ Images ကို ရွေးထားပြီး လက်ရှိ စမ်းလက်စတချို့ ရှိနေတာကို တွေ့ရပေမဲ့ ကိုယ့်ဆီမှာ မရှိသေးတာ ဖြစ်နိုင်ပါတယ်။

အဆင့် (၅)

Docker ကို ကိုယ့်ပရောဂျက်တွေ အတွက်သာ လေ့လာမယ်ဆိုရင် အများကြီး လေ့လာစရာတွေ ရှိပေမဲ့၊ n8n အသုံးပြုနိုင်ရုံ သက်သက်အတွက်တော့ အများကြီး မလိုပါဘူး။ သိသင့်တဲ့အခြေခံလေး လွတ်မသွားအောင် ဒီလို လိုတိုရှင်း မှတ်နိုင်ပါတယ်။



Container ထဲမှာ Image နဲ့ Volume ကို ထည့် Run ရပါတယ်။ Container ဟာ သီးခြားကွန်ပျူတာ မဟုတ်ပေမဲ့ သီးခြားကွန်ပျူတာ တစ်ခုကဲ့သို့ Run ပါတယ်။ Host လို့ခေါ်တဲ့ကိုယ့်ပင်မ ကွန်ပျူတာက ခွဲတမ်းပေးထားတဲ့ CPU တွေ RAM တွေနဲ့ အလုပ်လုပ်တာပါ။

Volume ဆိုတာ Host ကွန်ပျူတာထဲက ဖိုဒါတစ်ခုကို Container ထဲမှာ **C:** တို့ **D:** တို့လို Drive တစ်ခုကဲ့သို့ ချိတ်ပေးလိုက်တာပါ။ Linux System ဖြစ်လို့ **C:** တွေ **D:** တွေတော့ မသုံးပါဘူး။ စိတ်ကူးကြည့်လို့ရအောင် ပြောပြတာပါ။

Image ဆိုတာ ကိုယ်အဓိက Run လိုတဲ့ Software နဲ့ အဲဒီ Software အတွက် လိုအပ်တာတွေ ပေါင်းစပ်ပါဝင်တဲ့ အရာတစ်ခုပါ။ Image တစ်ခုကို Run ခိုင်းလိုက်ရင် Docker က Container တစ်ခု တည်ဆောက်ပြီး အဲဒီ Container ထဲမှာ Image ကို ထည့် Run ပေးလိုက်မှာ ဖြစ်ပါတယ်။

Volume ကတော့ လိုရင် ပူးတွဲအသုံးပြုနိုင်ပြီး။ မလိုရင်လည်း မသုံးဘဲနေလို့ရပါတယ်။

n8n with Docker Command

n8n ကို Docker နဲ့ Run တဲ့အခါ နည်းလမ်းနှစ်မျိုး ဖြစ်နိုင်ပါတယ်။ တစ်မျိုးက Command နဲ့ Run တာဖြစ်ပြီး နောက်တစ်မျိုးက Docker Desktop ထဲကနေ Run တာ ဖြစ်ပါတယ်။ နှစ်မျိုးလုံး ပြောပြပါမယ်။

Command နဲ့ Run ကြည့်ဖို့အတွက် Command Prompt (သို့) Terminal App ကို Admin အနေနဲ့ ဖွင့်လိုက်ပါ။ ပြီးတဲ့အခါ ဒီ Command ကို Run လိုက်ပါ။

```
docker volume create n8n_data
```

ဒါက n8n_data အမည်နဲ့ Volume တစ်ခု တည်ဆောက်လိုက်တာပါ။ ကိုယ်ဖန်တီးလိုက်တဲ့ Workflow တွေနဲ့ တခြားလိုအပ်တဲ့ အချက်အလက်တွေကို ဒီ Volume ထဲမှာ သိမ်းသွားစေဖို့ ဖြစ်ပါတယ်။

နောက်တစ်ဆင့်မှာ ဒီ Command ကို လေ့လာကြည့်ပါ။

```
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  -e GENERIC_TIMEZONE="Asia/Yangon" \
  -e TZ="Asia/Yangon" \
  -e N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true \
  -e N8N_RUNNERS_ENABLED=true \
  -v n8n_data:/home/node/.n8n \
  docker.n8n.io/n8nio/n8n
```

Linux တို့ Mac တို့မှာဆိုရင် အဲ့ဒီ Command ကို Run လိုက်ရင် ရပါတယ်။ Windows အသုံးပြုသူဆိုရင် အောက်ပါအတိုင်း Command တစ်ကြောင်းတည်းအနေနဲ့ Run ပေးလိုက်မှ အဆင်ပြေမှာ ဖြစ်ပါတယ်။

```
docker run -it --rm --name n8n -p 5678:5678 -e
GENERIC_TIMEZONE=Asia/Yangon -e TZ=Asia/Yangon -e
N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true -e
N8N_RUNNERS_ENABLED=true -v n8n_data:/home/node/.n8n
docker.n8n.io/n8nio/n8n
```

တော်တော်ရှည်တဲ့ Command ပါ။ ကူးရေးမဲ့အစား Copy ကူးယူအသုံးပြုလိုက်ပါ။ Asia/Yangon အစား ပြည်ပမှာ နေထိုင်သူဆိုရင် ကိုယ်နေထိုင်ရာ ဒေသရဲ့ Timezone အမှန်ကို ရှာပြီး အစားထိုးထည့်ပေးပါ။

အဓိပ္ပာယ်လေး နည်းနည်း ရှင်းပြချင်ပါတယ်။ `--name n8n` ဆိုတာ အခု Run မဲ့ Container ကို အမည်ပေးလိုက်တာပါ။ ကိုယ်က အမည်သတ်မှတ် မပေးရင် Docker က အမည်တစ်ခု သူ့ဘာသာ အလိုအလျောက် ပေးသွားမှာပါ။

`-p 5678:5678` ဆိုတာ ကိုယ့် Host ကွန်ပျူတာရဲ့ Port နံပတ် 5678 ကို Container ရဲ့ Port နံပတ် 5678 နဲ့ တွဲချိတ်ပေးလိုက်တာပါ။ ဒါကြောင့် ကိုယ့်စက်ထဲမှာပဲ 5678 ကို ခေါ်လိုက်ရင် Container ထဲက n8n Run ထားတဲ့ Port 5678 ကို ရောက်သွားမှာပါ။

`-v` ကို သုံးပြီး စောစောက ကြိုတင်ဖန်တီးထားတဲ့ Volume ကို ချိတ်သုံးထားပါတယ်။ နောက်ဆုံးမှာ n8n Image ဖိုင်ကို ထည့်ပေးထားပါတယ်။ Image ဖိုင်ကို အလိုအလျောက် Download ရယူပေးသွားမှာပါ။

Download ပြီးသွားအောင် စောင့်ဖို့တော့ လိုပါလိမ့်မယ်။ တစ်ကြိမ်ဒေါင်းပြီးရင် နောက်ပိုင်းမှာ ထပ်ဒေါင်းစရာ မလိုတော့ပါဘူး။ ပထမတစ်ကြိမ် ဒေါင်းပြီးသားကို သုံးပြီး အလုပ်လုပ်ပေးပါလိမ့်မယ်။

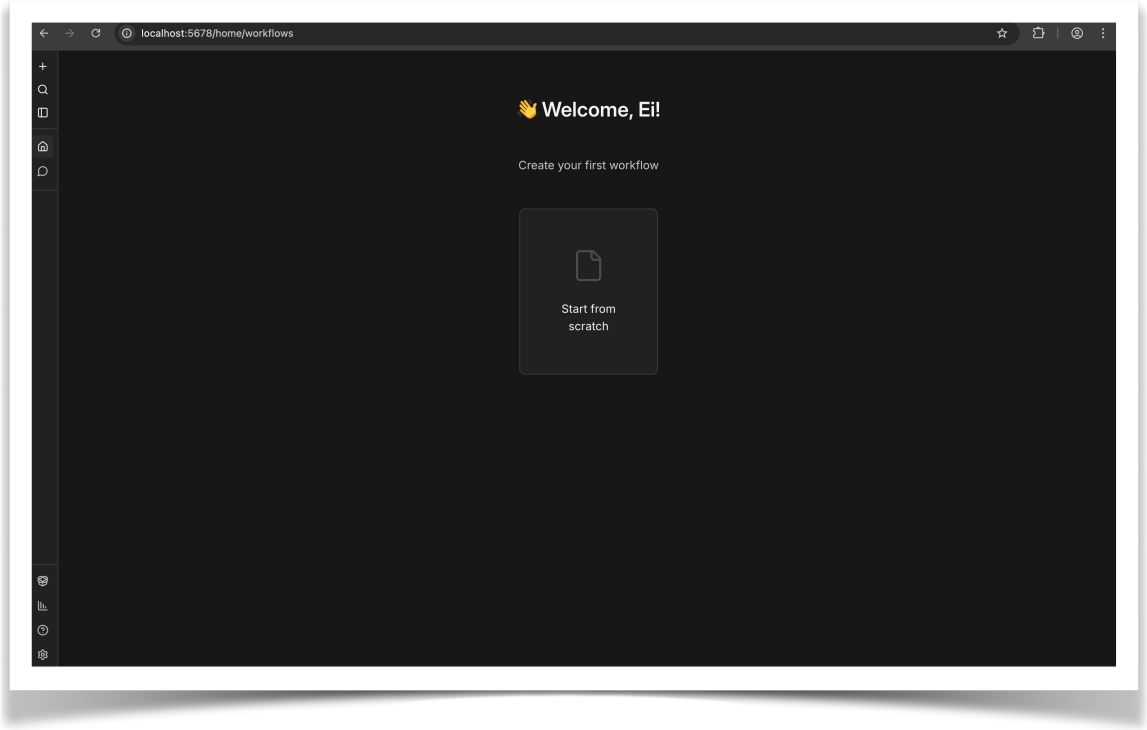
မူလ Command မှာ လိုင်းတစ်ခုချင်းစီရဲ့ နောက်ဆုံးက Back Slash (\ ) လေးတွေက၊ မ ပြီးသေးဘူး၊ နောက်တစ်လိုင်း ကျန်သေးတယ်ဆိုတဲ့ အဓိပ္ပာယ်ပါ။

Download ပြီးသွားတဲ့အခါ n8n ကို Run ပြီး အသင့်ဖြစ်နေပြီဖြစ်လို့ မိမိနှစ်သက်ရာ Web Browser မှာ `localhost:5678` လို့ ရိုက်ထည့်ပြီး ကြည့်လို့ရပါပြီ။

http://localhost:5678

အကောင့်တည်ဆောက်ခိုင်းတဲ့အခါ ဆောက်လိုက်ပါ။ နောက်ထပ် သူမေးတဲ့ မေးခွန်းလေးတချို့ကိုလည်း ဖြေပေးလိုက်ပါ။ ပုံတွေနဲ့ အတိအကျမပြောတာက အဲ့ဒီမေးခွန်းတွေ အဆင့်တွေက ပြောင်းနိုင်လို့ပါ။ ဖောင်လေးတွေ ဖြည့်ရတာပဲ မို့လို့ ကိုယ့်ဘာသာ ကြည့်ဖြည့်လိုက်ရင် အဆင်ပြေသွားလိမ့်မယ်လို့ ထင်ပါတယ်။

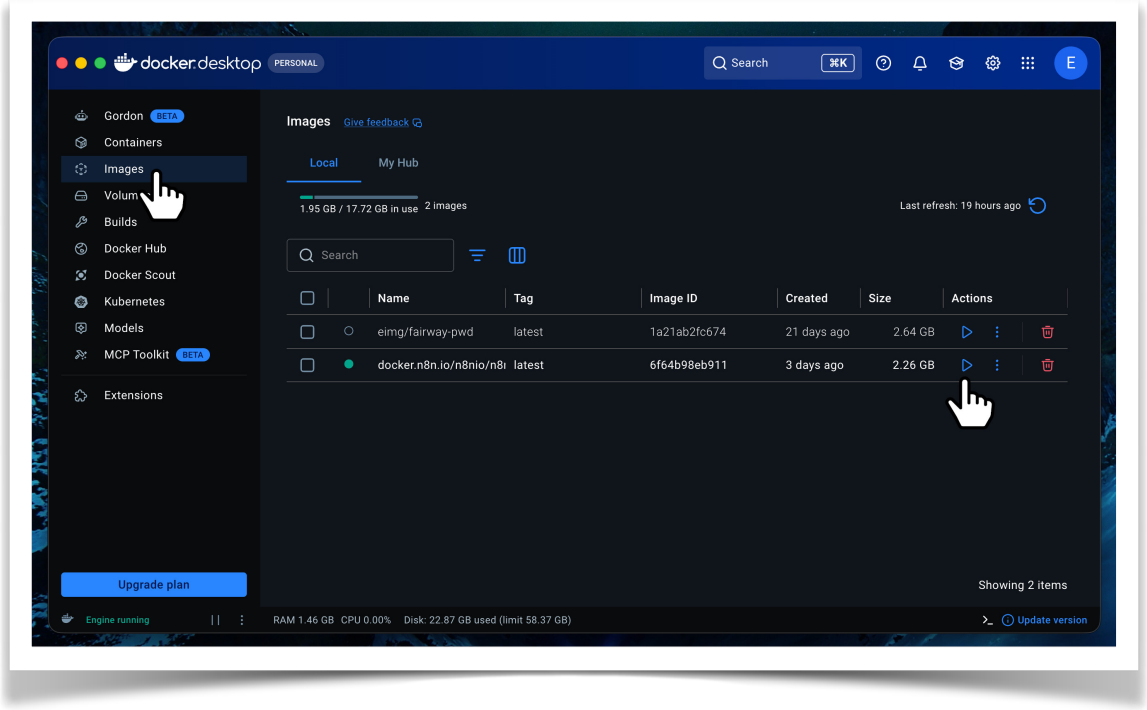
နောက်ဆုံး နမူနာပုံမှာ ပြထားသလို ရလဒ်ကို ရရှိလာပါလိမ့်မယ်။ ပေးထားတဲ့ ခလုတ်ကို နှိပ်ပြီး စတင်အသုံးပြုလို့ရပါပြီ။



n8n with Docker Desktop

n8n ကို Command နဲ့ Run တဲ့အခါ အဲ့ဒီ Command Run နေတဲ့ Terminal ကို အမြဲဖွင့်ထားဖို့ လိုပါတယ်။ Command ကို Ctrl+C နဲ့ ရပ်လိုက်ရင် (သို့မဟုတ်) Terminal ကို ပိတ်လိုက်ရင် n8n လည်း ပိတ်သွားမှာ ဖြစ်ပါတယ်။

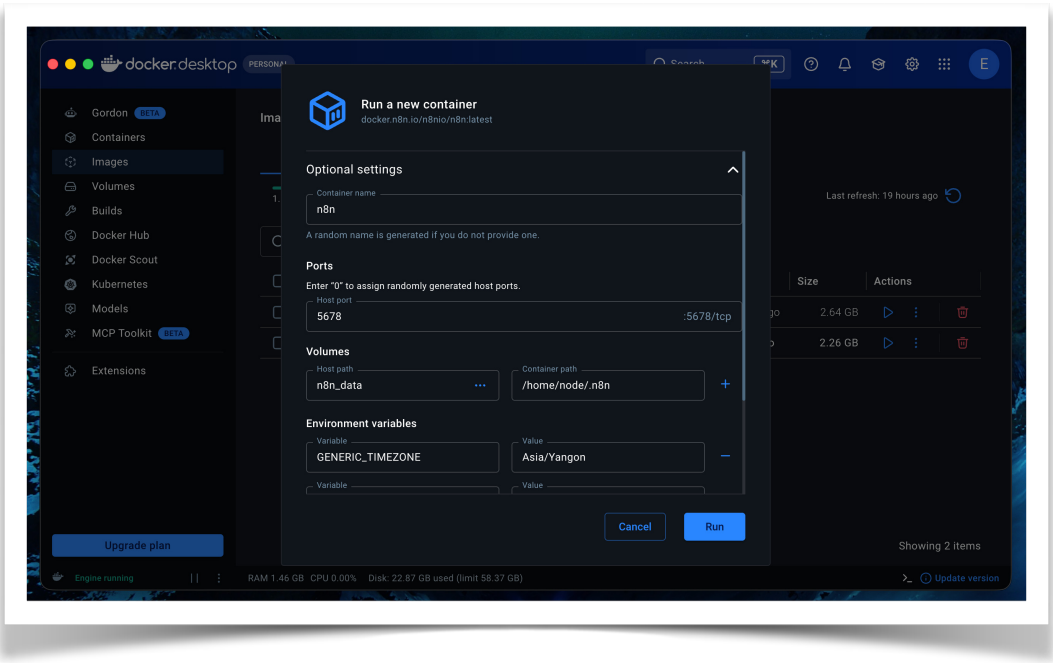
ဒါကြောင့် နောက်တစ်နည်းအနေနဲ့ Docker Desktop ကနေ Run နည်းကို ထည့်သွင်းဖော်ပြလိုက်ပါတယ်။



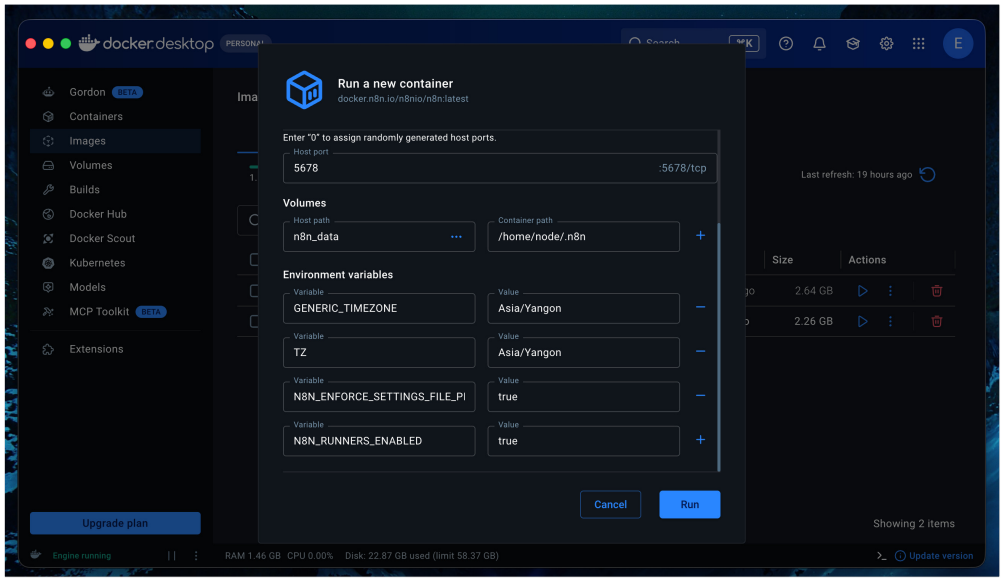
ပုံမှာ ပြထားသလို **Images** ကို နှိပ်ကြည့်လိုက်ပါ။ စောစောက Command ကိုသာ တစ်ကြိမ် Run ခဲ့ပြီး ဖြစ်မယ်ဆိုရင် စာဖတ်သူဆီမှာလည်း နမူနာပုံမှာ ပြထားသလို n8n Image ရှိပြီး ဖြစ်နေတာကို တွေ့ရပါလိမ့်မယ်။ Run ခလုတ်လေးကို နှိပ်ပြီး Run လိုက်ပါ။

Additional Settings ကို နှိပ်လိုက်ပါ။

ပြီးရင် နောက်တစ်ဆင့်ကပုံမှာ ပြထားသလို Container Name နေရာမှာ **n8n** လို့ ပေးလိုက်ပါ။ ပြီးရင် ပြထားသလို Port အတွက် **5678** နဲ့ Volume အတွက် ရှေ့က `n8n_data` နဲ့ နောက်က `/home/node/.n8n` ကို ထည့်ပေးလိုက်ပါ။

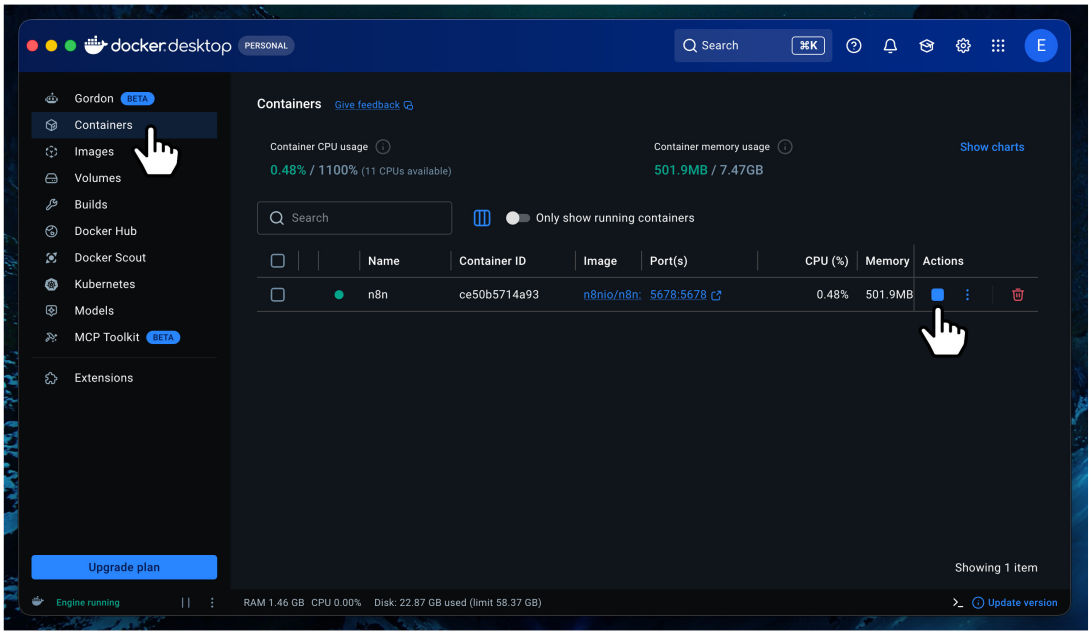


နောက်ပုံမှာဆက်ပြထားသလို Environment Variable တွေထပ်ထည့်ပေးရပါမယ်။



ဂရုစိုက်ကြည့်လိုက်ရင် စောစောက Command မှာ ပါတာတွေကိုပဲ ကူးယူပြီး ပြန်ထည့်ထားတယ်ဆိုတာ တွေ့ရပါလိမ့်မယ်။ အထက်ကပုံနှစ်ခုမှာ ပြထားသလို အချက်အလက်တွေ စုံပြီဆိုရင် Run လိုက်လို့ရပါပြီ။

ဒီနည်းမှာ တစ်ခုကောင်းတာက တစ်ကြိမ် Run ပြီးရင် နောက်တစ်ကြိမ် Run ချင်တဲ့အခါ Command တွေကူးရတာ၊ Variable တွေဖြည့်ရတာ ထပ်လုပ်စရာ မလိုတော့ဘဲ Docker Desktop ရဲ့ **Containers** စာရင်းကို သွားလိုက်ပြီး၊ စာရင်းထဲမှာ ရှိနေတဲ့ Container ကို ပြန် Run လိုက်ရုံပဲ ဖြစ်ပါတယ်။



စောစောကလိုပါပဲ။ Run ပြီးတဲ့အခါ နှစ်သက်ရာ Browser နဲ့ `localhost:5678` လို့ ရိုက်ထည့်ပြီး စမ်းကြည့်လို့ရသွားပြီပဲ ဖြစ်ပါတယ်။

http://localhost:5678

ဒီအထိ ရသွားရင် n8n ကို Docker နဲ့ ကိုယ့်စက်ထဲမှာ အောင်မြင်စွာ Run ပြီးသွားတာပါ။ မရခဲ့ရင်လည်း စိတ်မပူပါနဲ့။ နောက်အခန်းမှာ နောက်တစ်နည်းကို ဆက်လက်ဖော်ပြ ပေးသွားပါမယ်။

အခန်း (၂) - NPM နှင့် Install ပြုလုပ်ခြင်း

ဒီအခန်းမှာ n8n ကို ကိုယ့်စက်ထဲမှာ Install ပြုလုပ်နည်း နောက်တစ်နည်းကို ထပ်မံဖော်ပြသွားမှာ ဖြစ်ပါတယ်။ n8n ကို TypeScript လိုခေါ်တဲ့ Language တစ်မျိုးနဲ့ ရေးသားထားပြီး **NodeJS** လို့ခေါ်တဲ့ နည်းပညာနဲ့ Run လို့ ရပါတယ်။

ဒီနည်းကို အသုံးပြုဖို့အတွက် ပထမဆုံးအနေနဲ့ NodeJS ကို ဒီလိပ်စာမှာ Download ရယူပြီး Install လုပ်လိုက်ပါ။

https://nodejs.org/

လက်ရှိ ဒီစာကိုရေးသားနေစဉ်မှာ နောက်ဆုံးထွက်ရှိထားတဲ့ Version က Node 26 ဖြစ်ပါတယ်။ ဒါပေမယ့် နောက်ဆုံး ထွက်တာတွေထက် Long Term Support (LTS) ခေါ် ပိုတည်ငြိမ်တဲ့ Version ကို အသုံးပြုသင့်တဲ့ အတွက် **Node 24 LTS** ကို Download ရယူလိုက်ပါ။

သူကတော့ ရှုပ်ထွေးတဲ့ ပြဿနာတွေ တက်လေ့မရှိဘဲ အလွယ်တကူ Install လုပ်လို့ရ လေ့ရှိပါတယ်။ ဒါကြောင့် Install လုပ်ရမယ့် အဆင့်တွေကို ထည့်မပြောတော့ပါဘူး။ ကိုယ်တိုင်ပဲ Install ပြုလုပ်လိုက်ပါ။

Node ကို Install လုပ်ပြီးသွားရင် NPM လို့ခေါ်တဲ့ နည်းပညာတစ်မျိုးလည်း တစ်ခါတည်း ပူးတွဲပါဝင်သွားပါတယ်။ NPM ဆိုတာ Node Package Manager ခေါ် JavaScript တို့ TypeScript တို့နဲ့ ရေးသားဖန်တီးထားတဲ့ Package တွေကို Download ရယူပေးနိုင်တဲ့ နည်းပညာပါ။

NPM ကို အသုံးပြုပြီး n8n ကို Install လုပ်ဖို့အတွက် Command Prompt (သို့မဟုတ်) Terminal App ကို ဖွင့်လိုက်ပါ။ ပြီးတဲ့အခါ အခုလို Run လိုက်ပါ။

```
npm install n8n -g
```

နောက်ဆုံးက -g ရဲ့ အဓိပ္ပာယ်က Global လို့ခေါ်ပါတယ်။ တစ်ကြိမ် Install လုပ်ထား လိုက်ရင် ကြိုက်တဲ့နေရာကနေ ပြန်ခေါ်သုံးလို့ ရသွားတာပါ။

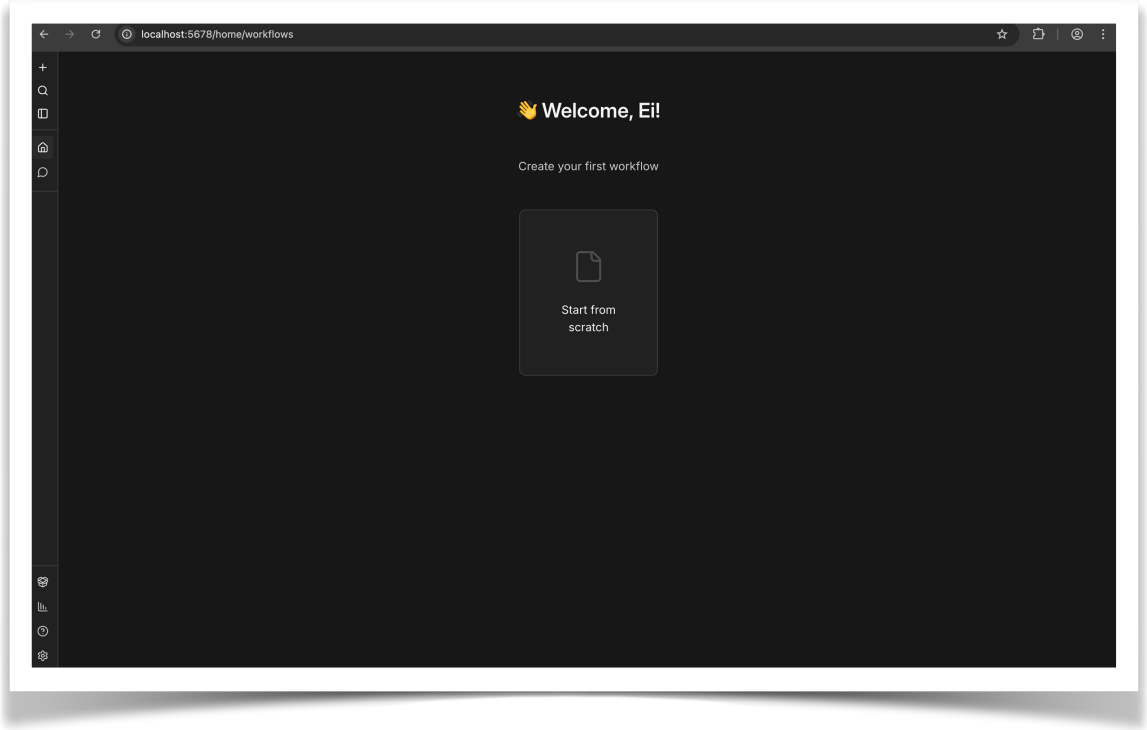
NPM က n8n နဲ့အတူ Dependencies ခေါ် ဆက်စပ်လိုအပ်တာတွေအကုန် တန်းစီပြီး Download လုပ်သွားမှာဖြစ်လို့ ခဏတော့စောင့်လိုက်ပါ။ အားလုံးပြီးသွားပြီဆိုရင် ဒီ Command နဲ့ Run ကြည့်လို့ရသွားပါပြီ။

```
n8n
```

ဒါပါပဲ။ ဘာမှထည့်စရာမလိုပါဘူး။ Timezone တွေဘာတွေ ထည့်ရင်ကောင်းပေမဲ့ လောလောဆယ် ဒီအတိုင်းပဲ စမ်းလိုက်ပါ။ နောက်ဆုံးခန်းကျတော့မှ ဒီ Command ကိုပဲ Timezone အပါအဝင် လိုအပ်တဲ့ Variable နဲ့ Run နည်း ထပ်ပြပေးပါမယ်။

Run ပြီးရင် Terminal ထဲမှာ Keyboard ကနေ O (အို) ကိုနှိပ်ပြီး Browser နဲ့ အလိုအလျောက် ဖွင့်ခိုင်းလို့ ရပါတယ်။ ဒါမှမဟုတ် မိမိနှစ်သက်ရာ Browser မှာ localhost:5678 လို့ ကိုယ့်ဘာသာ ရိုက်ထည့်ကြည့်လို့လည်း ရပါတယ်။

http://localhost:5678



n8n ကို Run လက်စရှိနေတဲ့ Terminal ကိုတော့ ဒီအတိုင်းဖွင့်ထားပေးရပါတယ်။ Run နေတာကို ရပ်လိုက်ချင်ရင်တော့ Ctrl + C ကို နှိပ်ပြီး ရပ်လို့ရပါတယ်။

ဒီနည်းကတော့ အများစုအတွက် Install လုပ်ရတာ ပိုအဆင်ပြေတဲ့နည်း ဖြစ်နိုင်ပါတယ်။ n8n Command ကို Run ထားတဲ့ Terminal ကို အမြဲဖွင့်ထားဖို့ လိုတယ်ဆိုတဲ့ အချက်နဲ့ တစ်ခါတစ်လေ လိုအပ်ရင် Restart လုပ်တဲ့အနေနဲ့ Ctrl+C နဲ့ရပ်ပြီး ပြန် Run ပေးဖို့ လို တတ်တယ်ဆိုတာကိုပဲ သတိပြုပါ။

Install လုပ်လို့ ရသွားပြီဆိုရင် Workflow နမူနာလေးတွေ လုပ်ကြည့်လို့ ရနေပါပြီ။ မ ကြည့်ခင် တခြားသိသင့်တာလေးတွေ နည်းနည်းလောက် ကြိုပြောလိုက်ချင်ပါတယ်။

အခန်း (၃) - JSON

n8n ကို အသုံးပြုဖို့အတွက် ကုဒ်တွေရေးဖို့ မလိုပေမဲ့ JSON လို့ခေါ်တဲ့ သဘောတရားလေး နားလည်ထားရင် အများကြီး အသုံးဝင်မှာပါ။ သူလည်းကုဒ်တစ်မျိုးပဲ ဆိုပေမဲ့ ဒီလောက်ကြီး မခက်ပါဘူး။ ဒီကုဒ်လေးကို လေ့လာကြည့်ပါ။

```
$json = {
  name: "Alice",
  age: 23,
  hobbies: ["Reading", "Music", "Yoga"]
}
```

JSON ဆို JavaScript Object Notation ရဲ့ အတိုကောက်ပါ။ ခပ်လွယ်လွယ် ပြောရရင် ဒေတာတွေပါတဲ့ Object တစ်မျိုး လို့ မှတ်နိုင်ပါတယ်။ အထက်က နမူနာမှာလို တွန့်ကွင်း အဖွင့်ပိတ်လေးနဲ့ ရေးရပါတယ်။ တွန့်ကွင်းထဲမှာ ဒေတာတွေရှိပါတယ်။ Property နဲ့ Value ပုံစံ အတွဲလိုက်လေးတွေ ရှိနေကြမှာပါ။

နမူနာအရ `name` Property ရဲ့ Value က "Alice" ပါ။ `age` Property ရဲ့ Value ကတော့ 23 ဖြစ်ပါတယ်။ တချို့တင်းကြပ်တဲ့စနစ်တွေမှာ Property တွေကို "Quote" အဖွင့်အပိတ်နဲ့ ရေးပေးရပါတယ်။ တချို့စနစ်တွေမှာတော့ နမူနာမှာလို Quote အဖွင့်အပိတ်တွေ မပါဘဲ ရေးတာကို လက်ခံကြပါတယ်။

Value တွေမှာ **Data Type** အမျိုးမျိုး ရှိနိုင်ပါတယ်။ နမူနာအရ "Alice" ဟာ **Text** ပါ။ String လို့ ခေါ်ပေမဲ့ Text လို့ပဲ မှတ်ထားရင် ရပါတယ်။ 23 ကတော့ **Number** ဖြစ်ပါတယ်။ Text တွေကို Quote အဖွင့်အပိတ်နဲ့ ရေးပြီး Number တွေကိုတော့ ဒီအတိုင်း Quote တွေမပါဘဲ ရေးပါတယ်။

ကုဒ်တွေရေးဖို့ မလိုဘူးလို့ ပြောပြီး ကုဒ်တွေအကြောင်း ထည့်ပြောနေလို့ စိတ်ညစ်မသွားပါနဲ့။ သိပ်မခက်ပါဘူး။ ဒီနေရာမှာ နည်းနည်းလေးအားစိုက်လိုက်ရင် နောက်ပိုင်း အများကြီး အသုံးဝင်သွားမှာ မို့လို့ပါ။

တခြား Data Type တွေလည်း ရှိသေးပေမဲ့ နားမျက်စိတွေ ရှုပ်မှာစိုးလို့ ထည့်မပြောပါဘူး။ အထက်က နမူနာမှာ ပါတဲ့ Array တစ်ခုကိုသာ ဖြည့်ကြည့်ပေးပါ။ Array ဆိုတာ List တစ်မျိုးပါပဲ။ မှတ်ရလွယ်အောင် List လို့ပဲ မှတ်ထားလည်း ရပါတယ်။

နမူနာအရ `hobbies` Property ရဲ့ Value က လေးထောင့်ကွင်းလေးနဲ့ ရေးထားတဲ့ List တစ်ခုပါ။ List ထဲမှာ Data တွေကို တန်းစီစုစည်းထားတာကို တွေ့ရပါလိမ့်မယ်။ နမူနာမှာ "Reading", "Music" နဲ့ "Yoga" ဆိုတဲ့ Text Item (၃) ခုပေးထားပါတယ်။

ဒီလို List ထဲက Item တွေကလည်း လိုအပ်ရင် Number တွေ၊ နောက်ထပ် List တွေ၊ နောက်ထပ် Object တွေ ဖြစ်နိုင်ပါသေးတယ်။

List ထဲက Item တွေမှာ ကိုယ်ပိုင်အခန်းနံပတ် **Index** လေးတွေ ရှိကြပါတယ်။ Zero ကနေ စပါတယ်။ ဒါကြောင့် `hobbies[0]` လို့ပြောရင် နမူနာဒေတာအရ "Reading" ကို ပြောနေတာပါ။ `hobbies[1]` လို့ပြောရင် "Music" ကို ပြောနေတာပါ။ Index တွေကို လေးထောင့်ကွင်း အဖွင့်အပိတ်ထဲမှာ ရေးရပါတယ်။

List ထဲက Item တွေကို Comma လေးနဲ့ ခွဲပြီး ရေးရသလိုပဲ ပင်မ Object ထဲက Property တွေကိုလည်း Comma လေးနဲ့ပဲ ခွဲခြားထားတာ သတိပြုပါ။ ဒါတွေကို ကိုယ်တိုင်ရေးတတ်ဖို့ မလိုပါဘူး။ မြင်တဲ့အခါ အဓိပ္ပာယ်ကိုသိရင် အဆင်ပြေပါတယ်။

Property တွေကို Dot သင်္ကေတနဲ့ ညွှန်းလို့ရပါတယ်။ ဥပမာ - `$json.name` လို့ပြောလိုက်ရင် "Alice" ကို ဆိုလိုပါတယ်။ `$json.age` ဆိုရင် 23 ကိုရမှာဖြစ်ပြီး `$json.hobbies[0]` ဆိုရင် "Reading" ကို ရမှာပဲ ဖြစ်ပါတယ်။

ဒီလောက်ပါပဲ။ တကယ်စိတ်ဝင်စားရင် လေ့လာစရာတွေ အများကြီး ရှိပေမဲ့ ဒီလောက်သိထားရင်ကို ရှေ့ဆက် အများကြီးအသုံးဝင်နေပါပြီ။

အခန်း (၄) - API

ဆက်လက်ပြီး API တွေရဲ့သဘောလေး အကျဉ်းချုပ် ထည့်ကြည့်သင့်ပါတယ်။ API ဆိုတာ Application Program Interface ရဲ့ အတိုကောက်ပါ။ တော်တော်လေး ကျယ်ပြန့်တဲ့နည်းပညာပါပဲ။ အသေးစိတ် သိချင်ရင် ကျွန်တော်ရေးသားထားတဲ့ API လိုတိုရှင်းစာအုပ်ကို အောက်ကလင့်မှာ အခမဲ့ရယူနိုင်ပါတယ်။

https://eimaung.com/api/

နောက်တော့မှ ရယူလေ့လာကြည့်ပါ။ အခုတော့ အဲ့ဒီလောက်ထိ မလိုသေးပါဘူး။ n8n ကို အသုံးပြုဖို့အတွက် အခြေခံသဘောလေးလောက် သိရင် လုံလောက်ပါတယ်။

API ဆိုတာ လှမ်းခေါ်လိုက်ရင် အလုပ်လုပ်ပေးတဲ့ နည်းပညာ လို့ ခပ်ရှင်းရှင်းသာ မှတ်လိုက်ပါ။ ကျွန်တော်တို့က API တွေ ဖန်တီးမှာ မဟုတ်ဘဲ API တွေကို ခေါ်သုံးမှာမို့လို့၊ ဒီနေရာမှာ ပိုအဓိကကျတာက ဖန်တီးနည်းမဟုတ်ပါဘူး၊ ဘယ်လိုခေါ်ရတာလဲဆိုတဲ့ ခေါ်ပုံခေါ်နည်းက အဓိက ဖြစ်ပါတယ်။

API Requests

ဒါလေးကိုအရင်ဆုံး လေ့လာကြည့်လိုက်ပါ။

```
Method: GET
URL: https://www.example.com
Headers: Accept: application/json
```

ဒီနမူနာက တကယ်အလုပ်မလုပ်ပါဘူး။ သဘောတရားကိုပဲ မြင်အောင် ကြည့်ရမှာပါ။

ထိပ်ဆုံးက **GET** ကို Request Method လို့ခေါ်ပါတယ်။ ဘာအတွက် လှမ်းခေါ်တာလဲ ဆိုတဲ့ ရည်ရွယ်ချက်ကို ဒီ Method နဲ့ ဖော်ပြပါတယ်။ နမူနာအရ GET ဖြစ်တဲ့အတွက် "လိုချင်တယ်" လို့ ပြောလိုက်တာပါပဲ။ ဘာကိုလိုချင်တာလဲ၊ ဘယ်ကနေလိုချင်တာလဲဆိုတာကို URL နဲ့ ပြောရပါတယ်။ နမူနာအရ www.example.com ကနေ လိုချင်တာပါ။

`Accept: application/json` ဆိုတာ ဒေတာတွေ ပြန်ပို့ပေးရင် JSON အနေနဲ့ ပို့ပေးပါလို့ ပြောလိုက်တာပါ။ ဒီလိုမျိုး ဖြည့်စွက် အသိပေးချင်တာတွေကို Headers အနေနဲ့ ပေးနိုင်ပါတယ်။

နောက်တစ်မျိုးထပ်ကြည့်လိုက်ပါ။

```
Method: GET
URL: https://www.example.com?user=alice
Headers: Accept: application/json
```

ဖွဲ့စည်းပုံ အတူတူပါပဲ။ ဒီတစ်ခါ ထူးခြားသွားတာက URL မှာ Question Mark (?) အမှတ်အသားလေးနဲ့ ဒေတာတချို့ထည့်ပေးထားပါတယ်။ ဒါကို **Query Parameter** လို့ ခေါ်ပါတယ်။ လှမ်းခေါ်ရင်းနဲ့ `user=alice` ဆိုတဲ့ အချက်အလက်ကို ပူးတွဲပေးပို့လိုက်တာပါ။ Query Parameter ကို ရည်ရွယ်ချက်အမျိုးမျိုးနဲ့ သုံးပါတယ်။ လှမ်းခေါ်ချိန်မှာ ထည့်ပေးတဲ့ ဒေတာတစ်မျိုးလို့ အလွယ်မှတ်နိုင်ပါတယ်။

နောက်တစ်ခုလောက် ထပ်ကြည့်လိုက်ပါဦး။

```
Method: POST
URL: https://www.example.com
Headers: Accept: application/json
Headers: Content-Type: application/json
Body: {
    "name": "Alice",
    "age": 23
}
```

ဒီတစ်ခါ GET မဟုတ်တော့ပါဘူး။ **POST** ဖြစ်သွားပါတယ်။ ဒါကြောင့် ခေါ်ယူတဲ့ ရည်ရွယ်ချက်က "လှမ်းပို့တယ်" ဆိုတဲ့ အဓိပ္ပာယ် ဖြစ်သွားပါတယ်။ ဘာကို လှမ်းပို့တာလဲဆိုတော့ အောက်ဆုံးက **Body** မှာပါတဲ့ ဒေတာကို လှမ်းပို့တာပါ။

ပို့လိုက်တာ JSON Data မှန်းသိအောင် Header မှာ Content-Type: application/json လို့ ထည့်ပြောထားပါတယ်။ ဒါကြောင့် Headers နှစ်ခုဖြစ်သွားပါတယ်။

ဒီနည်းနဲ့ Google Sheet API, Meta Graph API, X (Twitter) API, YouTube API စသည်ဖြင့် API နည်းပညာတွေကို လှမ်းခေါ်လို့၊ လှမ်းခိုင်းလို့ရတာပါ။ ဒီနေရာမှာ အိုင်ဒီယာရဖို့က အဓိကပါပဲ။ အသုံးပြုနည်းတွေကို သူ့နေရာနဲ့သူ ထပ်ပြောပြသွားမှာပါ။

GET နဲ့ POST နေရာမှာ နောက်ထပ် ဖြစ်နိုင်ခြေရှိတဲ့ Method တချို့ ရှိပါသေးတယ်။ PUT, PATCH နဲ့ DELETE တို့ကို အတွေ့ရများပါလိမ့်မယ်။ **PUT** ဆိုတာ တစ်ခုခုကို လှမ်းပြင်ခိုင်းလိုက်တာပါ။ **PATCH** ဆိုတာလည်း လှမ်းပြင်ခိုင်းလိုက်တာပါပဲ။

DELETE ဆိုရင်တော့ တစ်ခုခုကို လှမ်းဖျက်ခိုင်းလိုက်တဲ့ အဓိပ္ပာယ်ပါ။

API Responses

ဒီနည်းနဲ့ လှမ်းခေါ်၊ လှမ်းခိုင်းလိုက်တဲ့အခါ သက်ဆိုင်ရာ နည်းပညာက အလုပ်လုပ်ပြီးရင် ရလဒ်တွေ ပြန်ပေးပါတယ်။ ပြန်ပေးတဲ့ ရလဒ်တွေရဲ့ ဖွဲ့စည်းပုံက ဒီလို ဖြစ်နိုင်ပါတယ်။

```
Status: 200 OK
Body: {
    "api": "1.0",
    "status": "success"
}
```

ဒါကို **API Response** လို့ ခေါ်ပါတယ်။ API ကို လှမ်းခိုင်းလိုက်လို့ အလုပ်လုပ် ပြီးသွားတဲ့အခါ Status Code နဲ့ Body ဒေတာကို ပြန်ရတယ်လို့ အကျဉ်းချုပ် မှတ်နိုင်ပါတယ်။ Response မှာလည်း Headers တွေ ရှိပေမဲ့ ဒီနေရာမှာ လိုအပ်ချက် သိပ်မရှိလို့ ထည့်ပြမထားပါဘူး။

Status Code တွေထဲမှာ အတွေ့ရများမှာတွေက...

- **200** OK
- **400** Bad Request
- **401** Unauthorized
- **403** Forbidden
- **404** Not Found နဲ့
- **500** Internal Server Error တို့ ဖြစ်ပါတယ်။

200 OK ဆိုတာ အားလုံးအဆင်ပြေတယ်လို့ ပြောတာပါ။ 404 Not Found ကိုတော့ တွေ့ဖူးကြပါလိမ့်မယ်။ ကိုယ်က တစ်ခုခုလှမ်းခိုင်းလိုက်ပေမဲ့ အဲ့ဒါမရှိဘူးဆိုရင် သူက အလုပ်မလုပ်ဘဲ 404 Not Found လို့ ပြန်ပို့မှာပါ။

လှမ်းခိုင်းတဲ့အချိန်မှာ ကိုယ်ပေးပို့တာ နည်းမှားနေရင်လည်း သူက လက်ခံအလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ 400 Bad Request လို့ ပြန်လာပြောမှာပါ။

500 Internal Server Error ဆိုတာကတော့ Server မှာ Error တက်နေလို့ အလုပ်မလုပ်နိုင်သေးဘူးလို့ အကြောင်းပြန်တာပါ။ ကိုယ့်ဘက်က အမှား မဟုတ်ဘဲ၊ သက်ဆိုင်ရာ Server ဘက်မှာ တက်နေတဲ့ ပြဿနာပါ။

401 Unauthorized နဲ့ 403 Forbidden တို့ကိုတော့ လုပ်ခွင့်မရှိတာတွေ လှမ်းခိုင်းရင် ပြန်ရပါတယ်။ ဒီအချက်အလက်ကို ရယူခွင့်မရှိပါဘူး၊ ဒီအလုပ်ကို လုပ်ခွင့်မရှိဘူးဆိုတဲ့ သဘောနဲ့ ဒီ Code တွေကို ပြန်ပို့ပေးမှာပါ။

အသေးစိတ် မမှတ်နိုင်ရင်တောင်၊ ကုဒ်တစ်ခုချင်းစီမှာ သူ့အဓိပ္ပာယ်နဲ့သူရှိတယ်။ 200 ကုဒ်ကို ရမှ အရာရာအဆင်ပြေတယ်လို့ မှတ်နိုင်ပါတယ်။

API Auth

နောက်ထပ်ဖြည့်စွက်လေ့လာသင့်တာကတော့ **Authentication** နဲ့ **Authorization** တို့ပဲ ဖြစ်ပါတယ်။ တစ်ခုခု လှမ်းခိုင်းလိုက်တဲ့အခါ ကိုယ့်ဘက်က အထောက်အထားပြဖို့ လိုတာ ဖြစ်နိုင်ပါတယ်။

API Key လို့ခေါ်ပါတယ်။ **Access Token** လို့လည်း ခေါ်ပါတယ်။ အမျိုးမျိုးခေါ်ကြပေမဲ့ လိုရင်းကတော့အသုံးပြုခွင့်ကုဒ်ပါပဲ။ ဒီလို လက်တွေ့စမ်းကြည့်လိုက်ကြရအောင်။

Command Prompt (သို့မဟုတ်) Terminal ကို ဖွင့်ပြီး ဒီ Command လေး Run ကြည့်လိုက်ပါ။

```
curl -X GET https://api.nasa.gov/planetary/apod
```

ဒါဟာ Nasa API ကို လှမ်းခေါ်ပြီး Astronomy Picture of the Day ကို တောင်းယူလိုက်တာပါ။ အသုံးပြုတဲ့ Method က GET ဖြစ်ပြီး URL ကတော့ Nasa ဘက်က သူ့ Documentation မှာ သတ်မှတ်ထားတဲ့ URL အပြည့်အစုံကို သုံးထားပါတယ်။

အခုလိုပြန်လာပြောပါလိမ့်မယ်။

```
{
  "error": {
  "code": "API_KEY_MISSING",
  "message": "No api_key was supplied. …"
}
```

API Key မပါလို့ သုံးခွင့်မရှိဘူးလို့ ပြောနေတာပါ။ ဒါကြောင့် အောက်ကလင့်ကိုသွားပြီး ဖောင်လေးဖြည့်လိုက်ပါ။ API Key ကို အီးမေးလ်နဲ့ NASA က အခမဲ့ ပို့ပေးပါလိမ့်မယ်။

https://api.nasa.gov/

API Key ရပြီဆိုရင် အခုလို နောက်တစ်ခါပြန်စမ်းကြည့်လိုက်ပါ။

```
curl -X GET https://api.nasa.gov/planetary/apod?api_key=YOUR_KEY
```

URL မှာ API Key ထည့်ဖို့အတွက် Query Parameter ပါသွားပါတယ်။ **YOUR_KEY** နေရာမှာ ကိုယ့် API Key အမှန်ကို အစားထိုး ထည့်ပေးဖို့ လိုပါတယ်။

ဒေတာတချို့ JSON အနေနဲ့ ပြန်ပေးတယ်ဆိုတာကို တွေ့ရပါလိမ့်မယ်။ ဒီနေရာမှာ ဘာ ဒေတာ ပြန်ရလဲဆိုတာကို အဓိက မဟုတ်ပါဘူး။ တချို့လုပ်ဆောင်ချက်တွေက API Key ပါမှသာ လုပ်ခွင့်ရှိတယ်ဆိုတာကို လက်တွေ့စမ်းကြည့်လိုက်တာပါ။

API Key ပေးပို့နည်းတွေကတော့ အမျိုးမျိုးပါပဲ။ အခုလို Query Parameter အနေနဲ့ ပို့လို့ရသလို နောက်ထပ် နည်းလမ်းတစ်ခုက ဒီလိုပါ။

```
Method: POST
URL: https://www.example.com
Headers: Authorization: Bearer YOUR_KEY
```

ဒီတစ်ခါ API Key ကို Query Parameter အနေနဲ့ မပို့တော့ဘဲ Authorization Header အနေနဲ့ ပို့လိုက်တာပါ။ တခြားနည်းလမ်းတွေလည်း ရှိသေးပေမဲ့ ဒီနှစ်မျိုးလောက် သိထားရင် မဆိုးပါဘူး။

ဒီကုဒ်တွေ ကိုယ်တိုင်ရေးစရာ မလိုပါဘူး။ သက်ဆိုင်ရာအသုံးအနှုန်းတွေကို တွေ့တဲ့အခါ အဓိပ္ပာယ်ကို နားလည်ဖို့ပဲ လိုပါတယ်။

ဒီအကြောင်းတွေကို ကြိုပြောနေတာက နောက်အဆင့်တွေမှာ လွယ်သွားစေချင်လို့ပါ။ n8n ဟာ သိပ်မခက်လှတဲ့ နည်းပညာပါ။ ဒါလေးတွေ ကြိုသိမထားရင် မခက်သင့်ဘဲ ခက်နေနိုင်လို့ပါ။

အခန်း (၅) - First Workflow

ကြိုသိသင့်တာတွေ စုံပြီဖြစ်လို့ ပထမဆုံး Workflow လေးတစ်ခုလောက် စလုပ်ကြည့်ကြပါမယ်။ ကိုယ့်စက်ထဲမှာပဲ Setup လုပ်ပြီး လိုက်စမ်းကြည့်လို့ ရသလို၊ Setup မလုပ်ရသေးရင်လည်း n8n ပလက်ဖောင်းမှာပဲ နှစ်ပတ်အခမဲ့ရတဲ့ အကောင့်နဲ့ စမ်းကြည့်လိုက်ပါ။

https://n8n.io/

n8n ကိုယ်တိုက် သိပ်မခက်ပါဘူး။ ခက်နေတဲ့ အကြောင်းရင်းတွေထဲမှာ တစ်ခုက ဘာကို လုပ်ကြည့်ရမယ်မှန်း မသိတာဖြစ်နိုင်ပါတယ်။ ဒါကြောင့် Workflow တွေ Automation တွေ မစဉ်းစားခင်၊ ကိုယ်လုပ်ချင်တာ ဘာလဲဆိုတာကို အရင်ရှင်းဖို့လိုပါတယ်။

"Manual မလုပ်တတ်တဲ့အလုပ်ကို Automation သွားလုပ်ရင် ပိုအချိန်ကုန်ပါလိမ့်မယ်" တဲ့။ တချို့ အတွေ့အကြုံရှိသူများက ပြောကြပါတယ်။ ဒါကြောင့် ကိုယ်ဘာလုပ်ချင်တာလဲ ကိုယ့်ဘာသာရှင်းမယ်။ အဲ့ဒီအလုပ်ကို ကိုယ်ဘာသာ Manual လုပ်တတ်မယ်ဆိုရင်.. Workflow Automation တွေ ဖန်တီးရတာ မခက်ပါဘူး။

ဒီအခန်းမှာ နမူနာအနေနဲ့ ရာသီဥတု အချက်အလက် ရယူတဲ့ အလုပ်လေး တစ်ခုလောက် လုပ်ကြည့်ကြပါမယ်။ ရာသီဥတုဆိုတာ ဒီလောက်ကြီး စိတ်ဝင်စားဖို့ မကောင်းလှပါဘူး။ ဒါပေမဲ့ စဉ်းစားရလွယ်ပြီး လူတိုင်းသိတဲ့ကိစ္စဖြစ်လို့ နမူနာစမ်းလို့ ကောင်းပါတယ်။

ရာသီဥတုနဲ့ ပတ်သက်တဲ့ အချက်အလက်တွေ ရရှိဖို့အတွက် API Key တွေဘာတွေ မလိုဘဲ အခမဲ့ရတဲ့ **Open-Meteo** လို့ခေါ်တဲ့ နည်းပညာကို အသုံးပြုနိုင်ပါတယ်။ ပြဿနာက Open-Meteo ကနေ ရာသီဥတုအချက်အလက်တွေ လိုချင်ရင်၊ လိုချင်တဲ့ဒေသရဲ့ တည်နေရာ လတ္တီကျူးတွေ၊ လောင်ဂျီကျူးတွေ သိဖို့လိုပါတယ်။ အသုံးပြုနည်းကဒီလိုပါ။

https://api.open-meteo.com/v1/forecast?**latitude**=16.8661&**longitude**=96.1951

ရန်ကုန်မြို့ရဲ့ ရာသီဥတုအခြေအနေ သိချင်ရင် ရန်ကုန်မြို့ရဲ့ တည်နေရာ လတ္တီကျူးတွေ၊ လောင်ဂျီကျူးတွေ သိဖို့လိုပါတယ်။ သိပ်တော့ ပြဿနာမရှိပါဘူး။ Open-Meteo မှာ အဲ့ဒီလို အချက်အလက်တွေကို ပေးနိုင်တဲ့ API လည်း ရှိနေပါတယ်။ ဒီလိုသုံးရတာပါ။

https://geocoding-api.open-meteo.com/v1/search?**name**=Yangon

ဒီတော့၊ ရာသီဥတုအခြေအနေ ရယူတယ်ဆိုတဲ့ အလုပ်ရိုးရိုးလေးကိုပဲ Manual လုပ်မယ်ဆိုရင် အဆင့်တွေက ဒီလိုဖြစ်သွားမှာပါ။

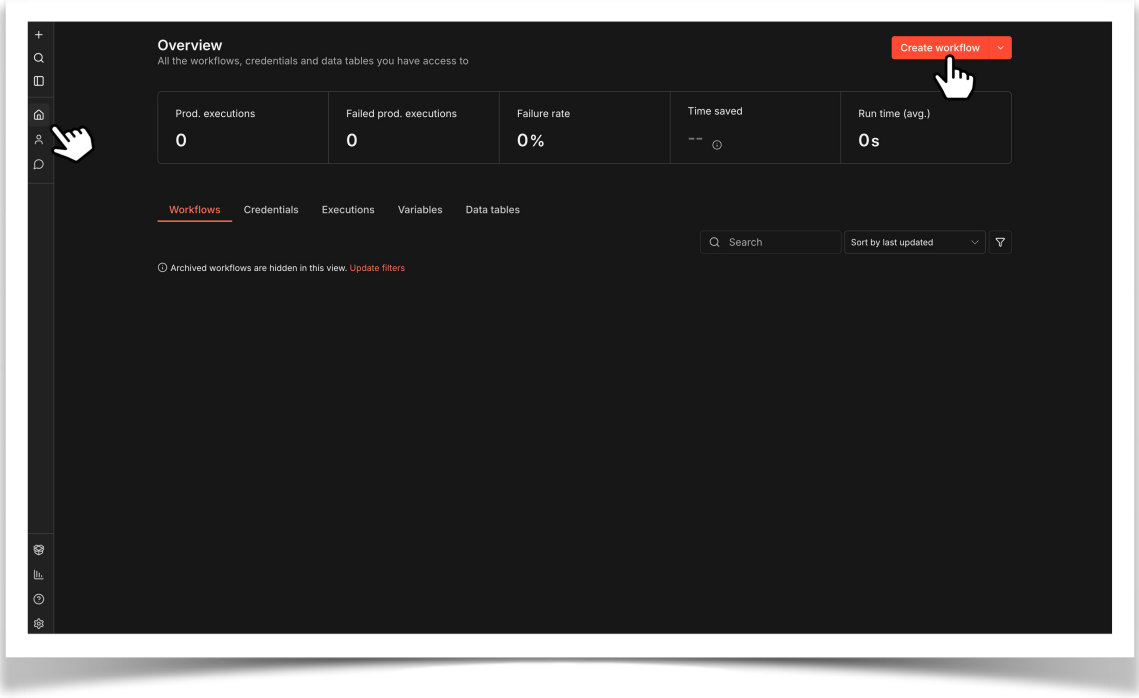
၁။ Yangon ရဲ့ တည်နေရာအချက်အလက်ကို ရှာရမယ်။

၂။ တည်နေရာဒေတာထဲက လတ္တီကျူး နဲ့ လောင်ဂျီကျူး ကိုယူရမယ်။

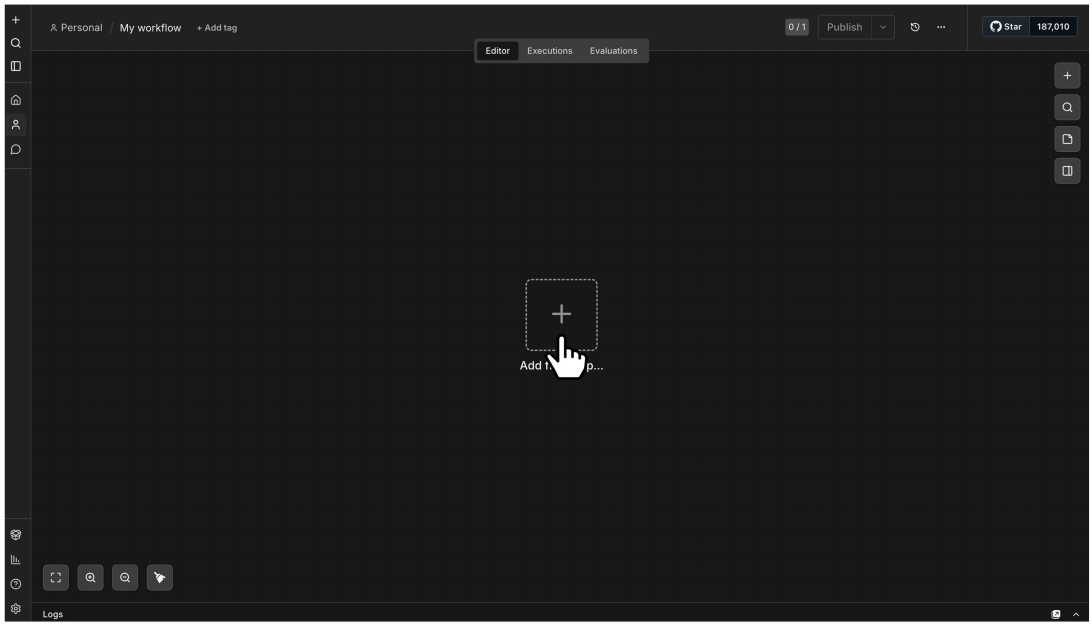
၃။ အဲ့ဒီ လတ္တီကျူး နဲ့ လောင်ဂျီကျူး ကိုသုံးပြီး ရာသီဥတုအခြေအနေ ယူရမယ်။

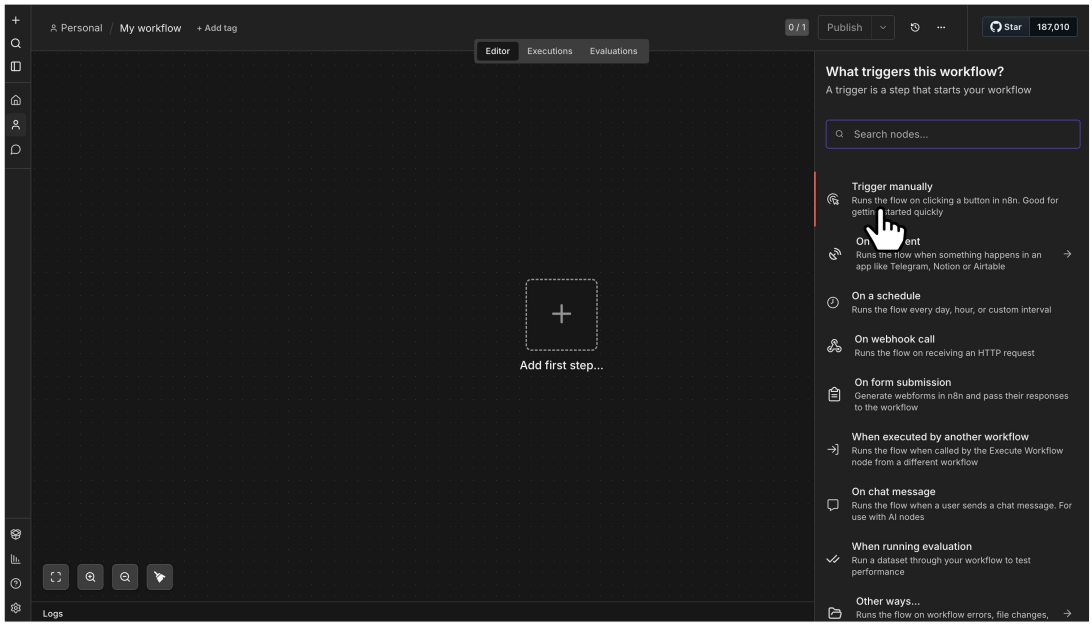
အခက်ကြီးတော့ မဟုတ်ပါဘူး။ ဒါပေမဲ့ Manual လုပ်မယ့်အစား အဲ့ဒီအလုပ်တွေ၊ အဆင့်တွေကို အလိုအလျောက် လုပ်သွားအောင် n8n နဲ့ Workflow တစ်ခု တည်ဆောက်ထားလိုက်လို့ ရတာပါ။

စတင်စမ်းသပ်ကြည့်နိုင်ဖို့အတွက် Create Workflow ခလုတ်ကိုနှိပ်ပြီး Workflow သစ်တစ်ခု ဖန်တီးလိုက်ပါ။



Canvas အလွတ်တစ်ခု ပေါ်လာတဲ့အခါ **Add first step...** ခလုတ်ကိုနှိပ်လိုက်ပါ။


Personal / My workflow + Add tag
0 / 1 Publish
Star 187,010
Editor Executions Evaluations
Logs


Personal / My workflow + Add tag
0 / 1 Publish
Star 187,010
Editor Executions Evaluations
Add first step...
What triggers this workflow?
A trigger is a step that starts your workflow
Search nodes...
Trigger manually
Runs the flow on clicking a button in n8n. Good for getting started quickly
On a schedule
Runs the flow every day, hour, or custom interval
On webhook call
Runs the flow on receiving an HTTP request
On form submission
Generate webforms in n8n and pass their responses to the workflow
When executed by another workflow
Runs the flow when called by the Execute Workflow node from a different workflow
On chat message
Runs the flow when a user sends a chat message. For use with AI nodes
When running evaluation
Run a dataset through your workflow to test performance
Other ways...
Runs the flow on workflow errors, file changes,
Logs

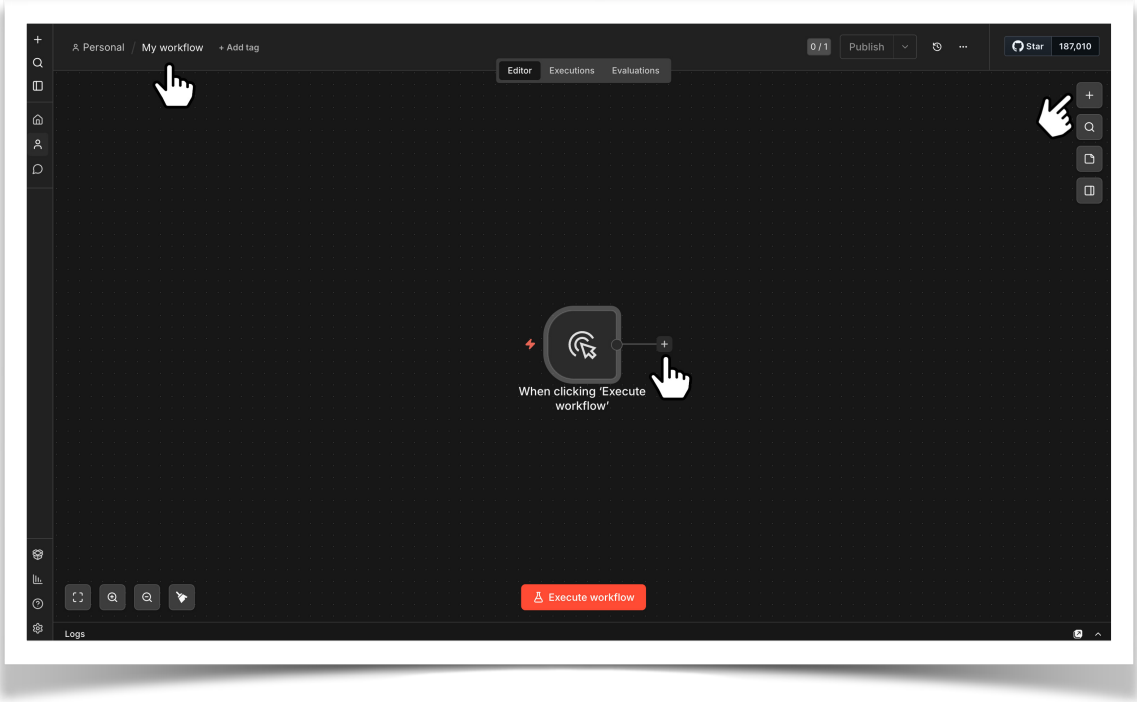
ပုံမှာပြထားသလို ညာဘက်ခြမ်းမှာ ရွေးချယ်စရာ Trigger Node စာရင်း ပေါ်လာပါလိမ့်မယ်။ Trigger Manually ကိုပဲ ရွေးလိုက်ပါ။ ပထမဆုံး Node လေးတစ်ခုကို စတင် ရရှိသွားပါလိမ့်မယ်။ n8n မှာ Node လေးတွေကို ချိတ်ဆက်ခြင်းအားဖြင့် လိုချင်တဲ့ Workflow ကို တည်ဆောက်ရတာပါ။

Node ဆိုတဲ့အသုံးအနှုန်းကို ပြီးခဲ့တဲ့အခန်းမှာ ပြောခဲ့တဲ့ NodeJS နဲ့ ရောမသွားပါနဲ့။ NodeJS ကိုလည်း Node လို့ပဲ သုံးကြပေမဲ့ မတူကြပါဘူး။ NodeJS ဆိုတာ ကုဒ်တွေ၊ ဆော့ဖ်ဝဲတွေ Run ပေးတဲ့ နည်းပညာတစ်မျိုးဖြစ်ပြီး အခုပြောနေတဲ့ Node ဆိုတာ Workflow မှာပါတဲ့ အစိတ်အပိုင်းလေးတစ်ခု ကို ဆိုလိုပါတယ်။

n8n မှာ Node အမျိုးအစား အမျိုးမျိုးရှိပါတယ်။ ပထမဆုံးတစ်ခုက အခုရွေးချယ်လိုက်တဲ့ Trigger Node ပါ။ Trigger Node ဆိုတာ Workflow ကို စတင်အသက်ဝင်စေတဲ့ စမှတ် ဖြစ်ပါတယ်။ Trigger Manually ကို ရွေးထားတဲ့အတွက် စမှတ်က Manual ဖြစ်ပါတယ်။ ကိုယ်တိုင်ခလုတ်နှိပ်ပြီး စမယ်ဆိုတဲ့ သဘောပါ။

Manual ခလုတ်နှိပ်ပြီးမစဘဲ၊ (၁) နာရီ တစ်ခါလုပ်ပါ၊ နေ့စဉ် မနက် (၉) နာရီမှာ လုပ်ပါ၊ စသည်ဖြင့် Schedule နဲ့ စလို့ရတဲ့ Node လည်းရှိပါတယ်။ အီးမေးလ်ဝင်လာရင် စရမယ်၊ ဖောင်ဖြည့်လိုက်ရင် စရမယ်၊ စတဲ့ ပြင်ပလုပ်ဆောင်ချက် တစ်ခုခု ဖြစ်ပေါ်တာနဲ့ အလိုအလျောက် စတဲ့ Node တွေလည်း ရှိပါတယ်။

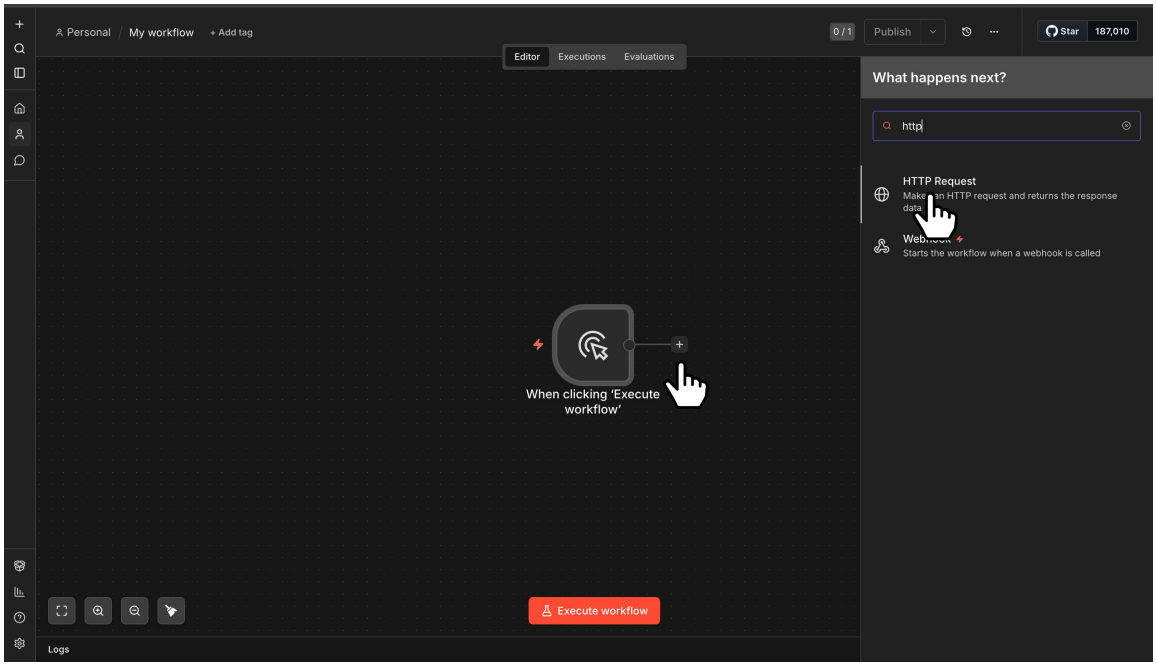
ဒီနေရာမှာ သိပ်အများကြီး ခေါင်းမစားပါနဲ့ဦး။ လက်တွေ့နမူနာနဲ့ ကြည့်သွားတာက ပိုရှင်းလို့ လုပ်စရာရှိတာကို ဆက်လုပ်ကြည့်ကြပါမယ်။



လက်ရှိအလုပ်လုပ်နေတဲ့ Workflow ရဲ့အမည်ကို ပြောင်းချင်ရင် ပုံမှာ ပြထားတဲ့ My workflow ဆိုတဲ့ ခေါင်းစဉ်လေးကို နှိပ်ပြီး ပြောင်းလို့ရပါတယ်။ နောက်ထပ် Node အသစ်တွေကို နည်းလမ်း နှစ်မျိုးနဲ့ ထည့်လို့ရပါတယ်။ လက်ရှိ Node ရဲ့ နောက်က **[+]** ခလုတ်လေးကို နှိပ်ပြီး သူနဲ့ ချိတ်ဆက်ထားတဲ့ Node တွေ ထပ်ထည့်လို့ရပါတယ်။

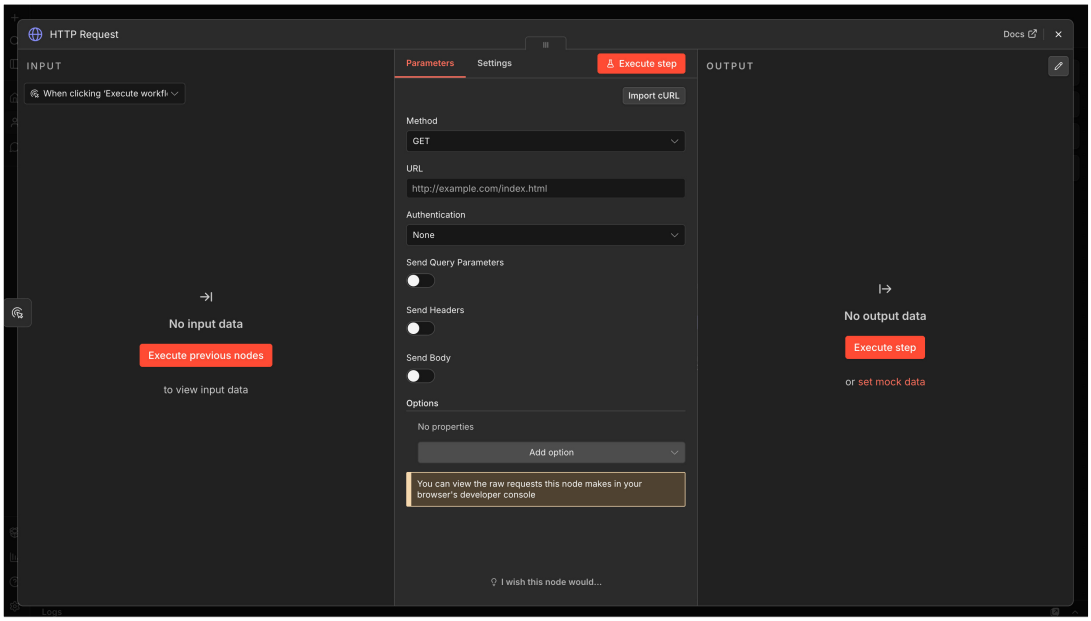
ဟိုးညာဘက်အပေါ်နားက **[+]** ခလုတ်လေးကို နှိပ်ပြီးတော့ ချိတ်ဆက်မထားတဲ့ သီးခြား Node တွေကို ထပ်ထည့်လို့ရပါတယ်။ ပြီးတော့မှာ ကိုယ့်ဘာသာ ချိတ်ဆက်ချင်တဲ့ Node တွေနဲ့ ဆွဲချိတ်လိုက်လို့ ရပါတယ်။ လိုအပ်ချက်ပေါ်မူတည်ပြီး နှစ်သက်ရာ ခလုတ်ကို နှိပ်နိုင်ပါတယ်။

လောလောဆယ် နမူနာအနေနဲ့ Manual Trigger Node ရဲ့နောက်က **[+]** ခလုတ်လေးကို နှိပ်ပြီး နောက်ထပ် Node တစ်ခု ထပ်ထည့်လိုက်ပါ။ ဒီတစ်ခါ ပေါ်လာတဲ့ Node စာရင်းထဲမှာ http လို့ ရိုက်ရှာလိုက်ပြီး **HTTP Request Node** ကို ရွေးလိုက်ပါ။



ထပ်တိုးလိုက်တဲ့ HTTP Request Node အတွက် Parameter တွေ သတ်မှတ်ပေးဖို့ နောက်တစ်ဆင့်က ပုံမှာပြထားသလို လာပြပါလိမ့်မယ်။ ဒီအဆင့်လေးက အရေးကြီးပါတယ်။ သေချာလိုက်ကြည့်လိုက်ပါ။

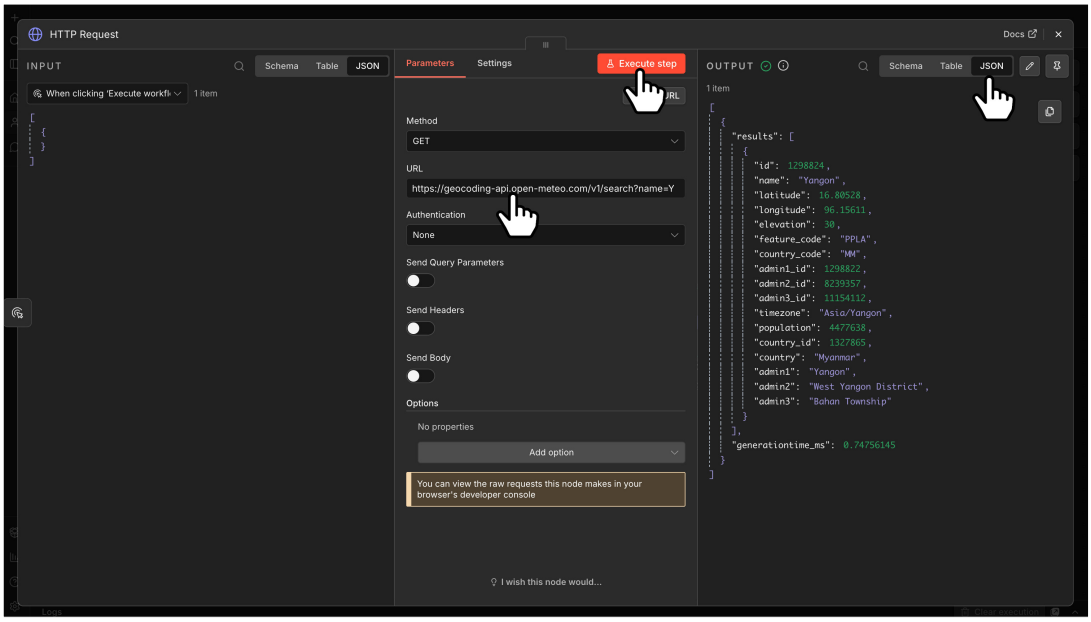
ဘယ်ဘက်ခြမ်းမှာ ရှေ့ Node ကနေရတဲ့ ဒေတာတွေ ရှိမှာဖြစ်ပါတယ်။ အဲ့ဒီ ဒေတာတွေက လက်ရှိ Node ရဲ့ Input အဝင်ဒေတာ ဖြစ်သွားမှာပါ။ လောလောဆယ် ဘာမှ မရှိသေးပါဘူး။



ဥာဘက်ခြမ်းမှာ ဒီ Node ကို Run လိုက်ရင်ရလာမဲ့ ရလဒ် Output ဒေတာတွေ ကို ပြပါတယ်။ လောလောဆယ် Execute မလုပ်ရသေးလို့ ဘာမှ မရှိသေးပါဘူး။ အလယ်မှာ လက်ရှိ Node အတွက် **Parameters** တွေ Settings တွေ သတ်မှတ်ပေးလို့ရပါတယ်။

နောက်တစ်ဆင့်က ပုံမှာပြထားသလို URL နေရာမှာ ဒီ URL လေး ကူးထည့်ပြီး Execute လုပ်ကြည့်လိုက်ပါ။ Method တို့ Authentication တို့လို ကျန်အချက်အလက်တွေ တစ်ခုမှ ပြောင်းစရာ မလိုသေးပါဘူး။

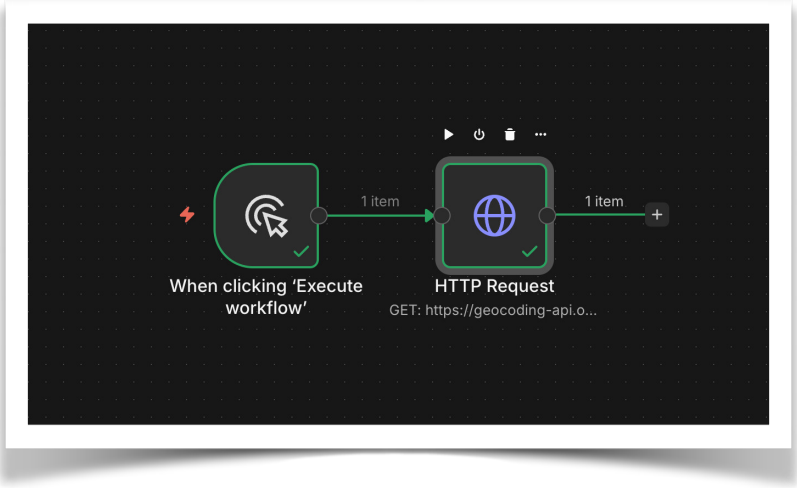
https://geocoding-api.open-meteo.com/v1/search?name=**Yangon**&count=1



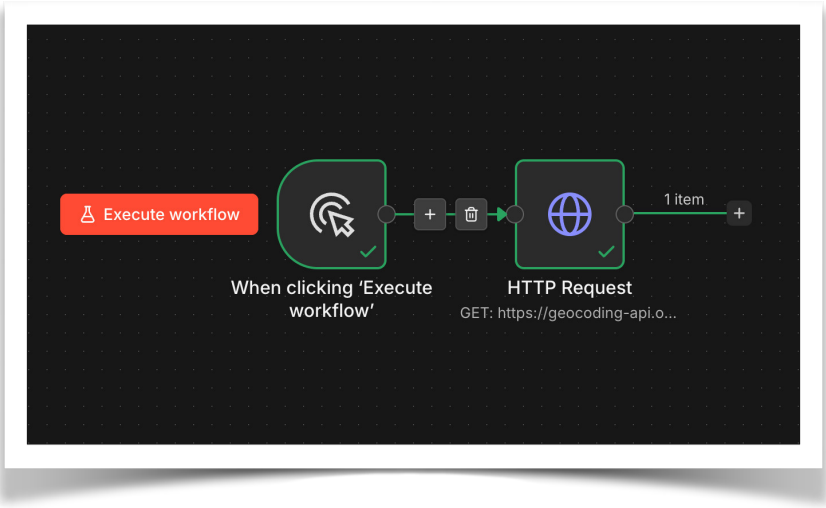
ညာဘက်ခြမ်းမှာ Open-Meteo API ကပြန်ပေးတဲ့ ရန်ကုန်မြို့ရဲ့ တည်နေရာ အချက်အလက်တွေ ပြန်ရတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။ အချက်အလက်တွေထဲမှာ **latitude** နဲ့ **longitude** တို့လည်း ပါပါတယ်။

နောက်ထပ်သတိပြုရမှာက ဖော်ပြပုံကို JSON လို့ရွေးထားပါတယ်။ လိုအပ်ရင် Schema တို့ Table တို့နဲ့လည်း ကြည့်လို့ရပါတယ်။ တချို့ဒေတာတွေက Table လေးနဲ့ကြည့်ရင် ပိုကောင်းပါတယ်။ လက်ရှိဒေတာကတော့ JSON အတိုင်းကြည့်တာ ပိုကောင်းလို့ JSON ကို ရွေးထားတာပါ။ ကိုယ်တိုင်ပဲ ကျန် Option တွေ ရွေးစမ်းကြည့်လိုက်ပါ။

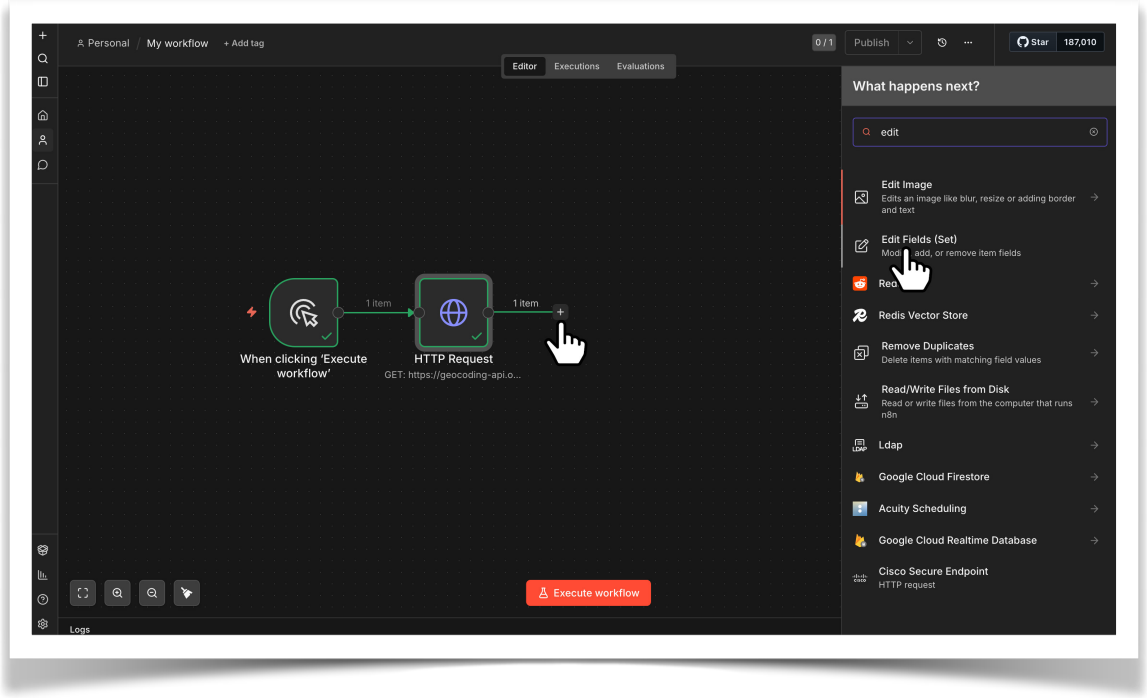
Parameter ထည့်ပြီးပြီ၊ စမ်းပြီးပြီဆိုရင် ပိတ်လိုက်ပါ။ Workflow ရဲ့ လက်ရှိအခြေအနေက ဒီလိုဖြစ်သွားပါပြီ။



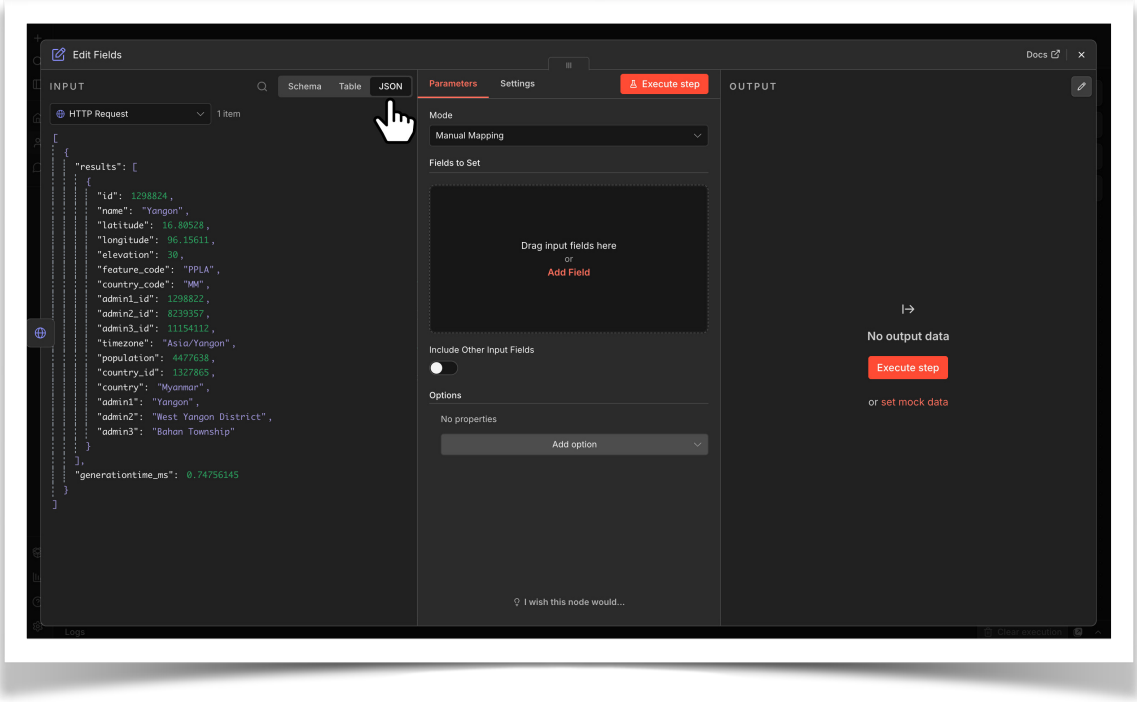
Node လေးတွေကို Mouse Pointer နဲ့ ထောက်လိုက်ရင် Run တဲ့ခလုတ်၊ ခဏပိတ်ထားလို့ရတဲ့ခလုတ်နဲ့ ဖျက်တဲ့ခလုတ်တွေကို တွေ့ရမှာ ဖြစ်ပါတယ်။ နောက်ဆုံးက ... Menu ခလုတ်ကို နှိပ်ပြီး Node ကို Rename လုပ်ပြီး အမည်ပြောင်းတာတွေ လုပ်လို့ရပါတယ်။



Node နှစ်ခုကြားက Connection လိုင်းလေးကိုထောက်ကြည့်လိုက်ရင်တော့ ကြားထဲမှာ နောက်ထပ် Node တွေ ထပ်ထည့်ချင်ရင် ထည့်လို့ရတဲ့ ခလုတ်နဲ့၊ Connection ကို ပြန်ဖျက်လို့ရတဲ့ ခလုတ်တွေကို တွေ့ရမှာပါ။



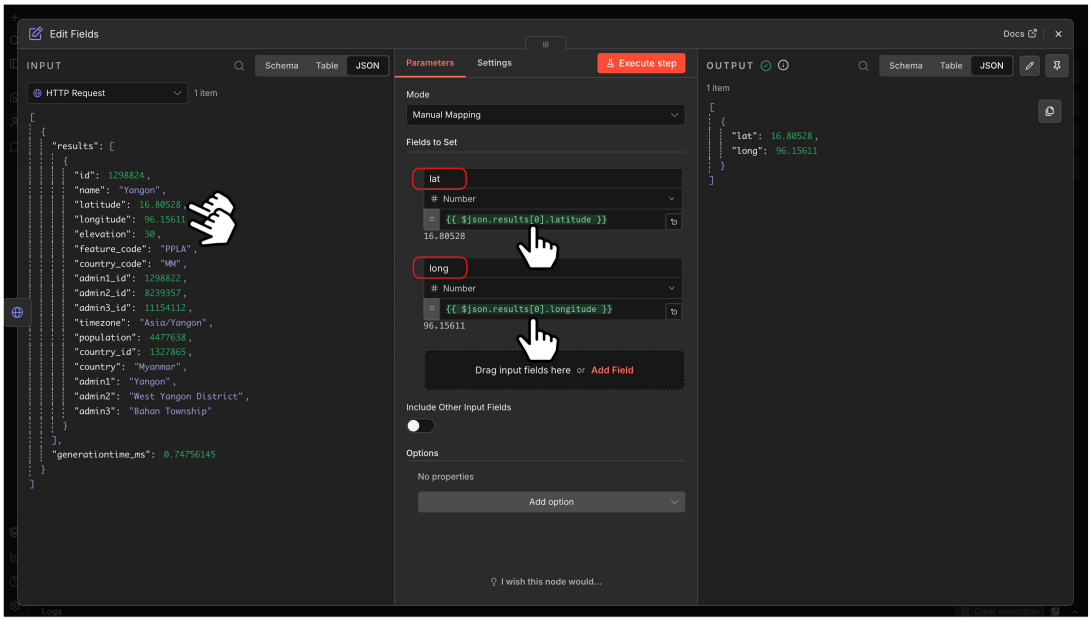
နောက်တစ်ဆင့်အနေနဲ့ ပုံမှာပြထားသလို HTTP Request Node နောက်က **[+]** လေးကို နှိပ်ပြီး၊ ပေါ်လာတဲ့ Node စာရင်းထဲကနေ edit ကို ရိုက်ရှာလိုက်ပါ။ **Edit Fields (Set)** ဆိုတဲ့ Node ကို ရွေးလိုက်ပါ။



ပေါ်လာတဲ့ရလဒ်ရဲ့ ဘယ်ဘက်ခြမ်းမှာ လက်ရှိ HTTP Request ကနေ ရရှိထားတဲ့ ဒေတာတွေကို တွေ့မြင်ရပါလိမ့်မယ်။ အဲဒီ ဒေတာတွေကိုလည်း JSON View နဲ့ ကြည့်ထားတာ ထပ်ဆင့်သတိပြုပါ။

အလယ်မှာ **Drag input fields here** ဆိုတဲ့ ဧရိယာလေးကို တွေ့ပါလိမ့်မယ်။ ဘယ်ဘက်ခြမ်းမှာ ရှိနေတဲ့ ဒေတာတွေထဲက **latitude** ကို Mouse Pointer နဲ့ပဲ Drag & Drop ဆွဲပြီး ထည့်လိုက်ပါ။

ပြီးတဲ့အခါ **longitude** ကိုလည်း နောက်တစ်ကြိမ် ထပ်ဆွဲထည့်လိုက်ပါ။ အခုလိုရလဒ်ကို ရရှိပါလိမ့်မယ်။



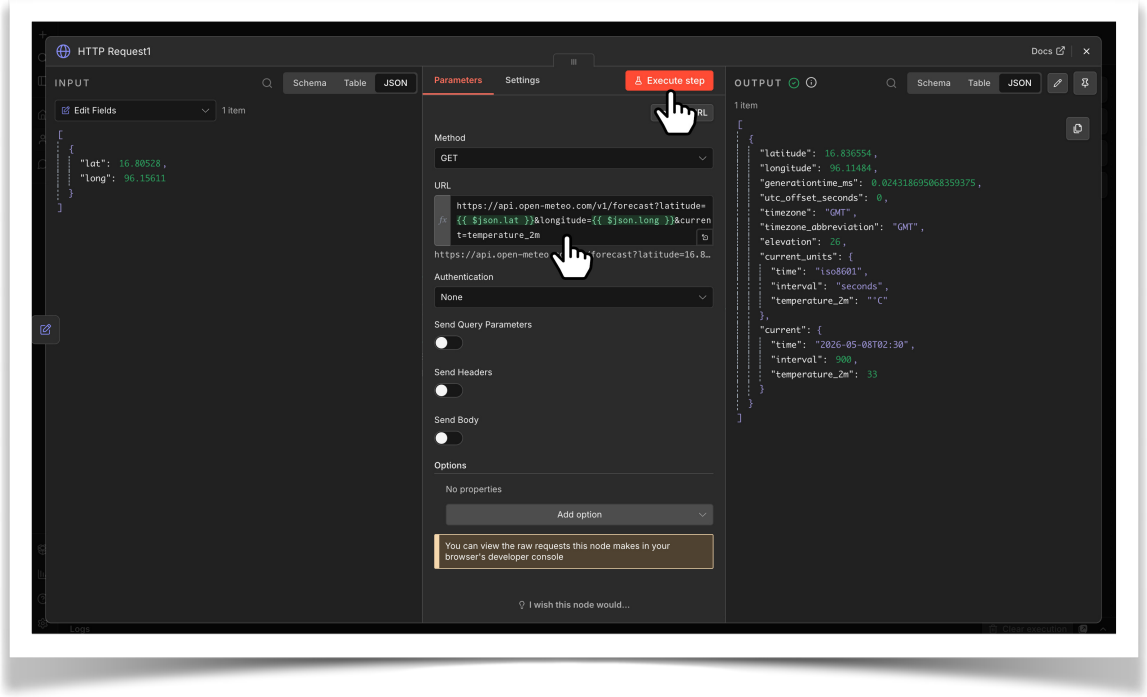
ကုဒ်တွေမြင်ပြီး လန့်မသွားပါနဲ့။ ကိုယ်တိုင်ရေးစရာမလိုပါဘူး။ ကိုယ်က လိုချင်တဲ့ဒေတာကို ရွေးပြီး ဆွဲထည့်လိုက်တာနဲ့ အလိုအလျောက် ရေးထည့်ပေးသွားတာပါ။ ဒါကြောင့် အဓိပ္ပာယ်ကို ဖတ်ရှုနားလည်ရင် တော်တော်အဆင်ပြေနေပါပြီ။ ဒေတာတွေ အများကြီး ရှိတဲ့ထဲက **latitude** နဲ့ **longitude** နှစ်ခုတည်းကို ဆွဲထုတ်ယူလိုက်တဲ့ သဘောပါ။

ဒေတာရဲ့အမည်က၊ `result[0].latitude` တို့ `result[0].longitude` တွေဖြစ်နေလို့ တိုတိုရှင်းရှင်း `lat` နဲ့ `long` လို့ပဲပေးထားတာကို အနီရောင်ဝိုင်းပြထားတဲ့ နေရာမှာ သတိပြုပါ။ ပြီးရင် Execute လုပ်ကြည့်လိုက်တဲ့အခါ ညာဘက်ခြမ်းမှာ ရထားတဲ့ ဒေတာတွေအများကြီးထဲက လတ္တီကျူးနဲ့ လောင်ဂျီကျူး နှစ်ခုတည်းကိုပဲ lat, long ဆိုပြီး ပြန်ရတာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

ပြန်ပိတ်လိုက်လို့ရပါပြီ။ နောက်တစ်ဆင့်မှာ နောက်ထပ် HTTP Request Node တစ်ခုကို ထပ်ထည့်လိုက်ပြီး ဒီ URL အတိုင်း အတိအကျ ကူးထည့်ပေးလိုက်ပါ။

```
https://api.open-meteo.com/v1/forecast?
latitude={{ $json.lat }}&longitude={{ $json.long }}
&current=temperature_2m,relative_humidity_2m,weather_code
```

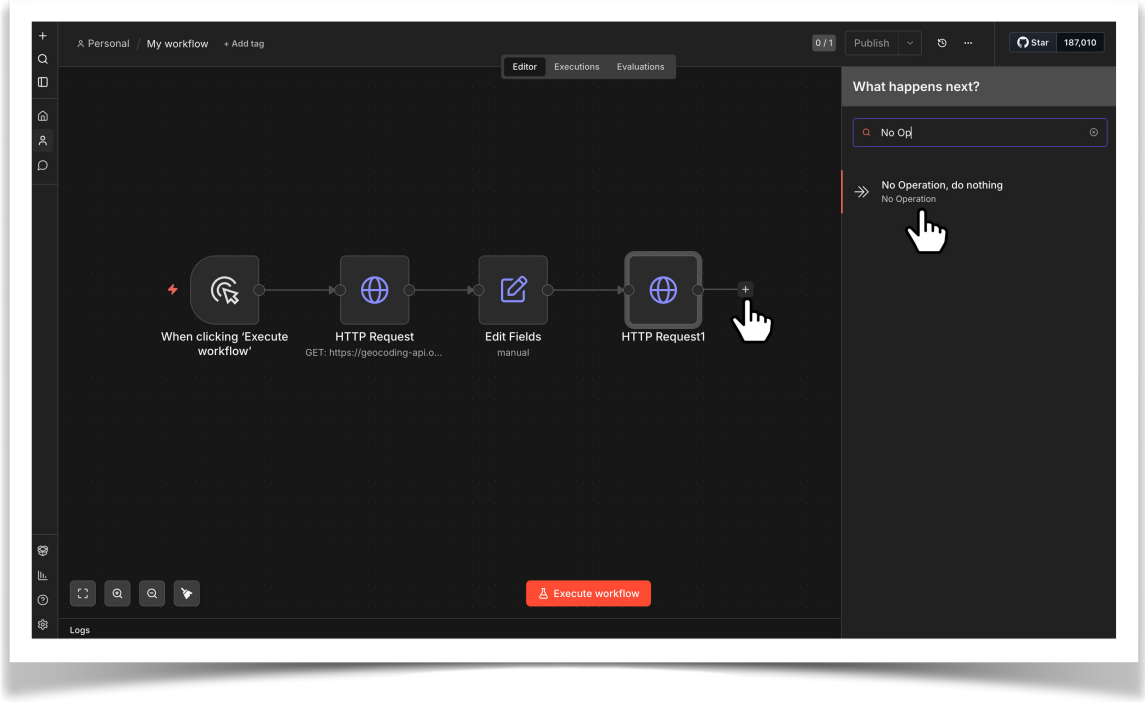
Open-Meteo ရဲ့ API URL ဖြစ်ပါတယ်။ ထူးခြားချက်အနေနဲ့ **latitude** နဲ့ **longitude** က အသေမဟုတ်တော့ဘဲ ရထားတဲ့ JSON Input ဒေတာရဲ့ lat နဲ့ long ကို အသုံးပြုလိုက်တာ ဖြစ်သွားပါတယ်။



URL ဟာ ရိုးရိုးစာမဟုတ်တော့ဘဲ ကုဒ်တွေပါတဲ့ Expression ဖြစ်သွားပါတယ်။

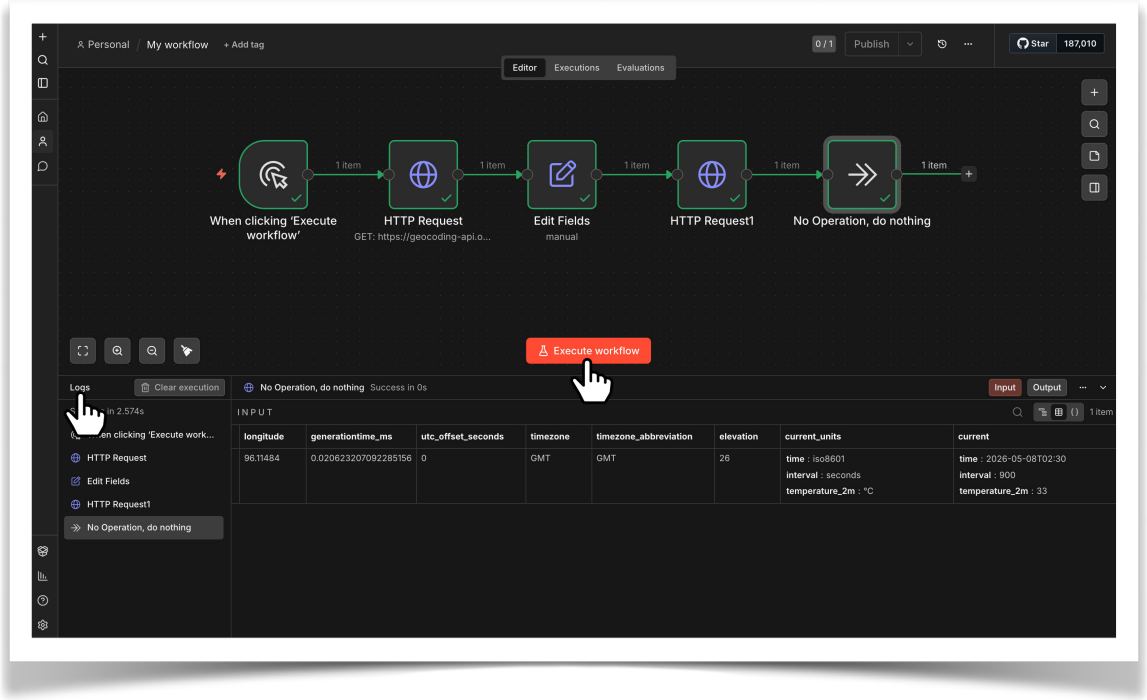
Execute လုပ်ကြည့်လိုက်ရင် ညာဘက်ခြမ်းမှာ ရန်ကုန်မြို့ရဲ့ လက်ရှိအပူချိန်အပါအဝင် အချက်အလက်တွေကို ရရှိတာ တွေ့ရပါလိမ့်မယ်။ နေပူ၊ မိုးရွာ စတဲ့ အချက်အလက်တွေ လိုချင်ရင် ရပေမဲ့ ထည့်ယူမထားပါဘူး။ အပူချိန်ကိုပဲ အဓိက ယူထားပါတယ်။

တစ်ခုပြီးတစ်ခု အဆင့်လိုက် ချိတ်ဆက်အလုပ်လုပ်သွားတဲ့ Workflow လေးတစ်ခုကို ရသွားပါပြီ။ **Execute Workflow** ခလုတ်ကိုနှိပ်ပြီး အစအဆုံး Run ကြည့်လို့ ရနေပါပြီ။



နောက်ဆုံးတစ်ခုအနေနဲ့ **No Operation Node** ကို ရှာထည့်လိုက်ပါ။ No Operation ဆိုတဲ့အတိုင်း ဘာအလုပ်မှ ဆက်မလုပ်ပါဘူး။ ရည်ရွယ်ချက်ကတော့ ဘာတွေအလုပ်လုပ်သွားလဲ အဆင့်လိုက်ကြည့်တဲ့အခါ နောက်ဆုံး ဒီနေရာမှာ အပြီးသတ်သွားတယ် ဆိုတာကို ထင်ရှားစေချင်လို့ပါ။

ပုံမှာပြထားသလို အောက်ခြေနားက **Log** ကိုနှိပ်ပြီး Workflow ကို Run ကြည့်လိုက်ရင် ဘယ်အဆင့်မှာ ဘာလုပ်သွားလဲ၊ ဘယ်လိုဒေတာတွေရလဲဆိုတာကို အသေးစိတ် တွေ့မြင်ရနိုင်ပါတယ်။

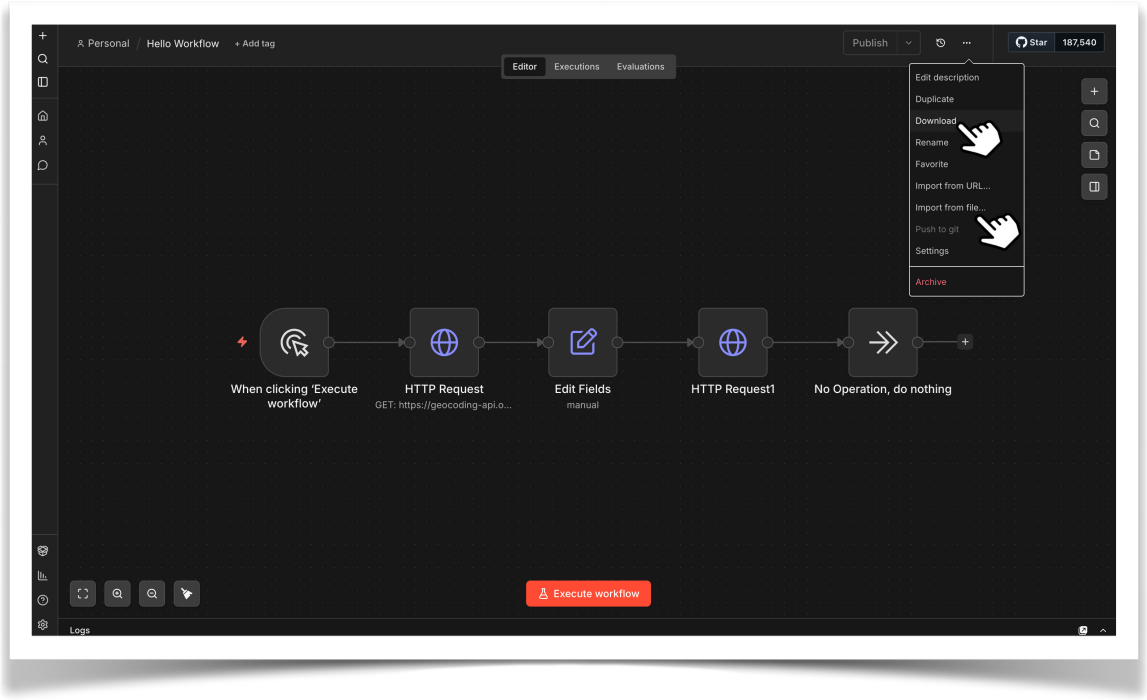


ဒီနေရာမှာလည်း ဒေတာတွေကို Table, JSON စသည်ဖြင့် အမျိုးမျိုး ကြည့်လို့ရပါတယ်။ စမ်းကြည့်လိုက်ပါ။ Execute Workflow နဲ့ Run ပြီး တစ်ဆင့်ချင်း ရလဒ်တွေကို လိုက်လေ့လာလိုက်ပါ။

ဒါဟာ ကျွန်တော်တို့ရဲ့ ပထမဆုံး အလိုအလျောက် အလုပ်လုပ်တဲ့ Workflow လေးတစ်ခု ကို n8n နဲ့ အောင်မြင်စွာ ဖန်တီးလိုက်နိုင်တာပဲ ဖြစ်ပါတယ်။

Download and Import

ကိုယ်လုပ်ထားတဲ့ n8n Workflow တွေကို ဖိုင်အနေနဲ့ **Download** ယူထားလို့ရပါတယ်။ အဲဒီ ဖိုင်တွေကို လိုအပ်တဲ့အခါမှာ ပြန်ပြီးတော့ **Import** လုပ်ယူလို့လည်း ရပါတယ်။



ဒီစာအုပ်မှာ ဖော်ပြထားတဲ့ နမူနာ Workflow တွေရဲ့ Download ဖိုင်တွေကို အောက်ပါလင့်မှာ ရယူနိုင်ပါတယ်။

https://github.com/eimg/n8n-workflows/archive/refs/heads/main.zip

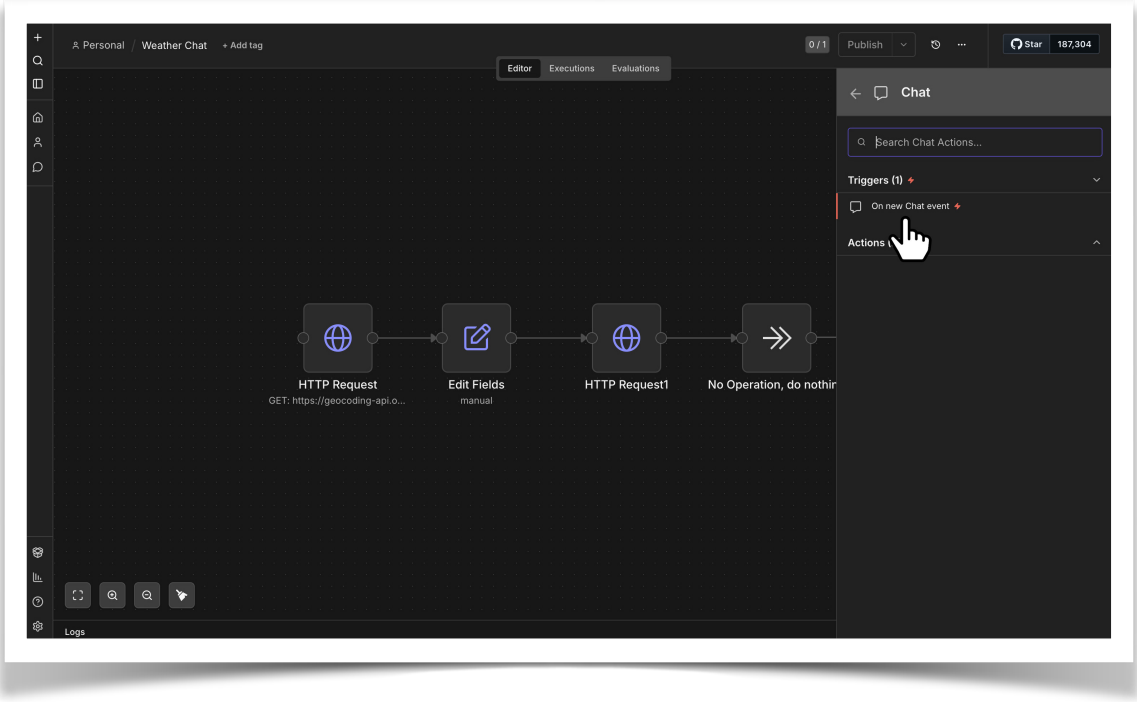
အကယ်၍ ကိုယ်တိုင်လိုက်လုပ်ကြည့်တာ တစ်ခုခု အဆင်မပြေခဲ့ရင်၊ ဒီဖိုင်တွေကို Import လုပ်ပြီးတော့လည်း စမ်းကြည့်လို့ ရပါတယ်။

အခန်း (၆) - Weather Chat

Workflow တစ်ခုကို ဖန်တီးတဲ့အခါ ဒေတာတွေကို လိုသလို ပေါင်းစပ်ရတာတွေ၊ ခွဲထုတ်ရတာတွေ လုပ်ရလေ့ရှိပါတယ်။ ပြီးတဲ့အခါ အခြေအနေပေါ်မူတည်ပြီး လုပ်ရတဲ့ အလုပ်တွေလည်း ရှိကြပါတယ်။ အဲ့ဒီလိုအလုပ်တွေကို လုပ်ပေးနိုင်တဲ့ Node တွေကို လေ့လာချင်တဲ့အတွက် ဒီအခန်းမှာလုပ်လက်စ Workflow ကို ပြင်ဆင်ဖြည့်စွက်ကြပါမယ်။

ပထမဆုံးအနေနဲ့ ပြီးခဲ့တဲ့အခန်းတုံးက လုပ်ခဲ့တဲ့ Workflow ကို Duplicate လုပ်ပြီး ပွားယူလိုက်ပါ။ ပြီးခဲ့တဲ့အခန်းမှာ ပြောခဲ့တဲ့ **Download** တို့ **Import from file...** တို့နဲ့အတူ **Duplicate** လည်း ရှိပါတယ်။ ကြည့်လိုက်ပါ။ ရလာတဲ့ Workflow မိတ္တူကို ကျွန်တော်ကတော့ Weather Chat လို့ အမည်ပေးလိုက်ပါတယ်။

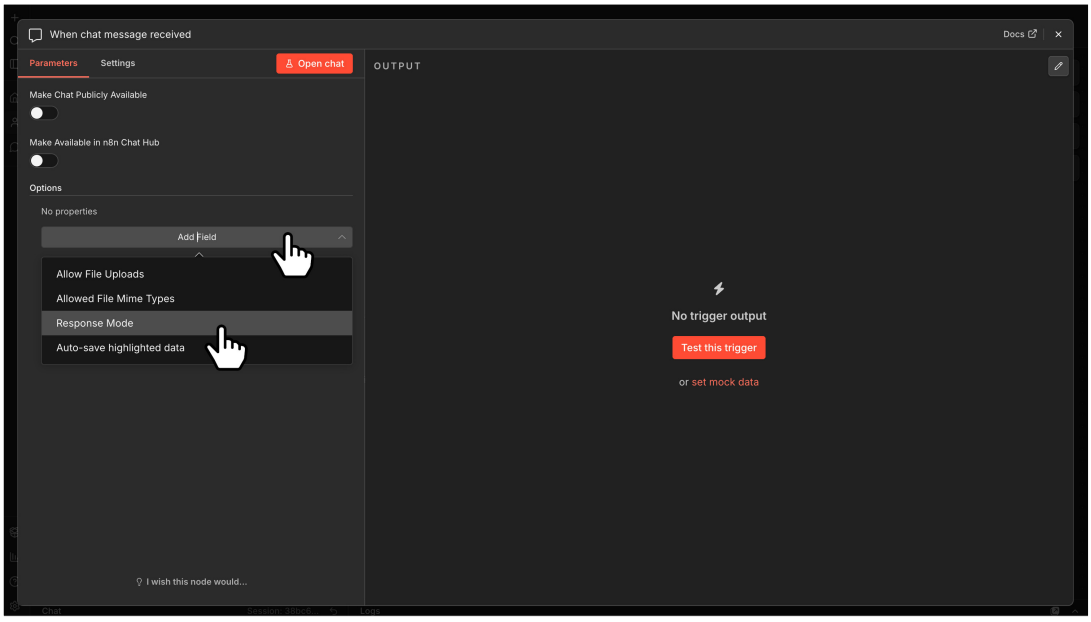
လိုအပ်တဲ့ ပြင်ဆင်မှုတွေ လုပ်ဖို့အတွက် ရှေ့ဆုံးက **Manual Trigger Node** ကို ဖျက်လိုက်ပါ။ Click နှိပ်ရွေးပြီး Keyboard ကနေ Delete နှိပ်လိုက်ရင် ရသလို၊ Mouse Pointer နဲ့ Node ကိုထောက်ပြီး ပေါ်လာတဲ့ ခလုတ်လေးတွေထဲက Delete ကို ရွေးလိုက်ရင်လည်း ရပါတယ်။

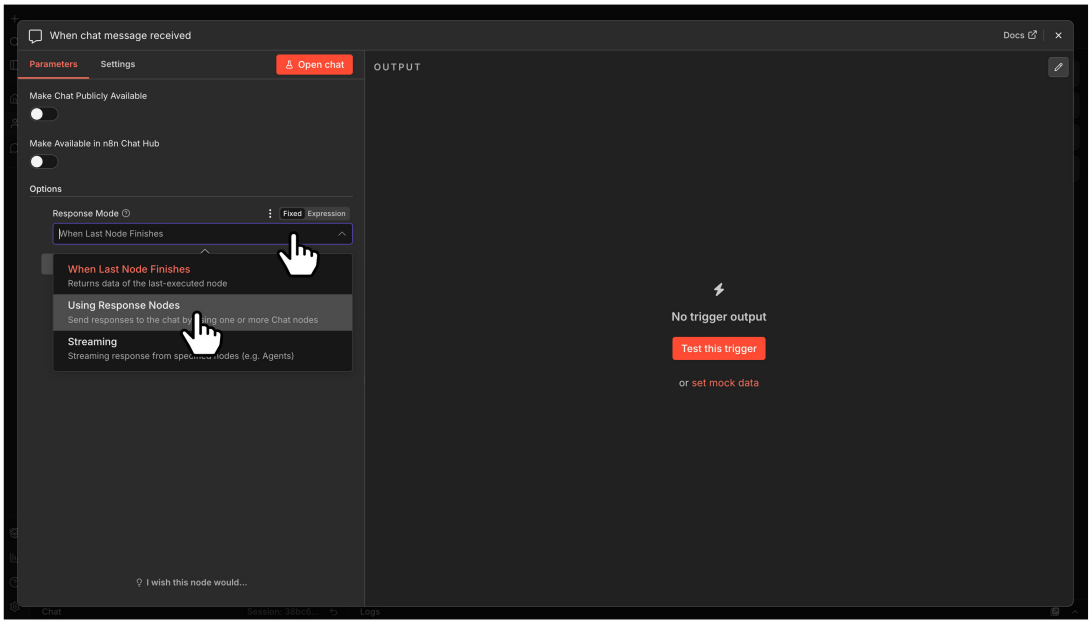


နောက်ဆုံး Node ရဲ့ဘေးက **[+]** မနှိပ်ဘဲ၊ ဟိုးညာဘက်ခြမ်း ထိပ်နားက **[+]** ခလုတ်လေးကို နှိပ်ပြီး Chat ကို ရိုက်ရှာလိုက်ပါ။ ပြီးတဲ့အခါ ပုံမှာပြထားသလို **On new Chat event** ဆိုတဲ့ Node ကို ရွေးလိုက်ပါ။

ဒါဟာ Chat Trigger Node ပါပဲ။ Workflow ကို Manual စမှာ မဟုတ်တော့ဘဲ Chat Message တစ်ခု ဝင်လာတဲ့အခါ စမယ်ဆိုတဲ့ အဓိပ္ပာယ်ပါ။

ပေါ်လာတဲ့ Parameters တွေထဲက ပုံမှာပြထားသလို **Add Field** ခလုတ်ကိုနှိပ်ပြီး **Response Mode** ကို နှိပ်ပါ။

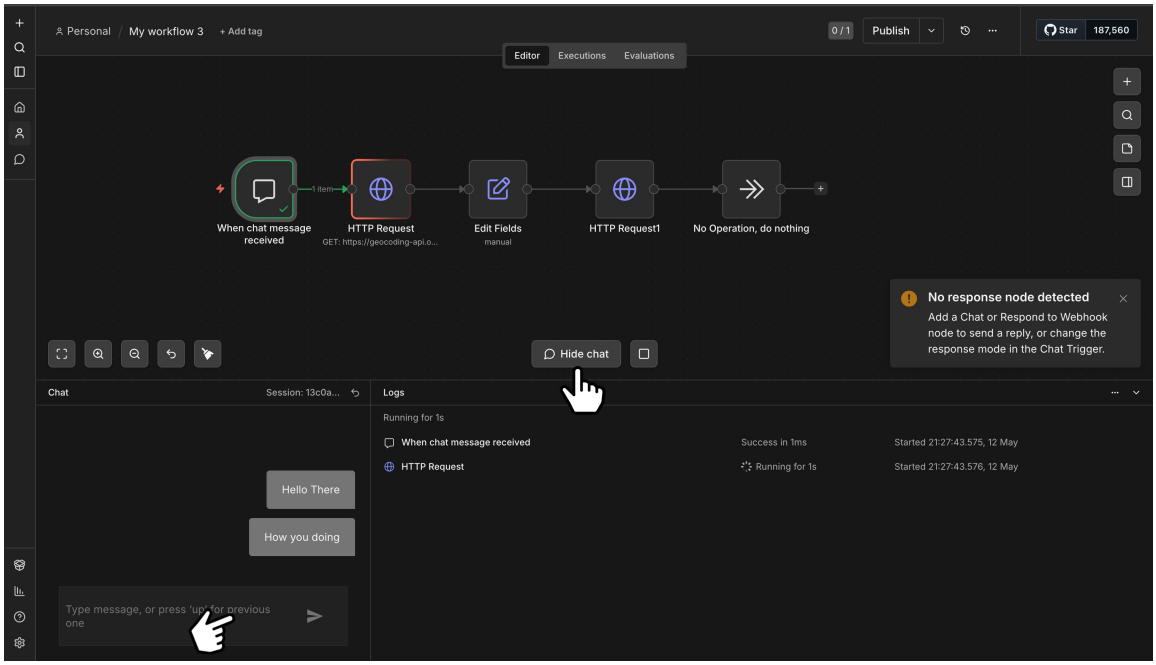

When chat message received
Docs
Parameters
Settings
Open chat
OUTPUT
Make Chat Publicly Available
Make Available in n8n Chat Hub
Options
No properties
Add Field
Allow File Uploads
Allowed File Mime Types
Response Mode
Auto-save highlighted data
No trigger output
Test this trigger
or set mock data
I wish this node would...
Chat
Logs


When chat message received
Docs
Parameters
Settings
Open chat
OUTPUT
Make Chat Publicly Available
Make Available in n8n Chat Hub
Options
Response Mode
Fixed
Expression
When Last Node Finishes
When Last Node Finishes
Returns data of the last-executed node
Using Response Nodes
Streaming
No trigger output
Test this trigger
or set mock data
I wish this node would...
Chat
Logs

ပြီးတဲ့အခါ နောက်တစ်ပုံမှာ ပြထားသလို **Using Respond Nodes** ကို ထပ်ရွေးပါ။

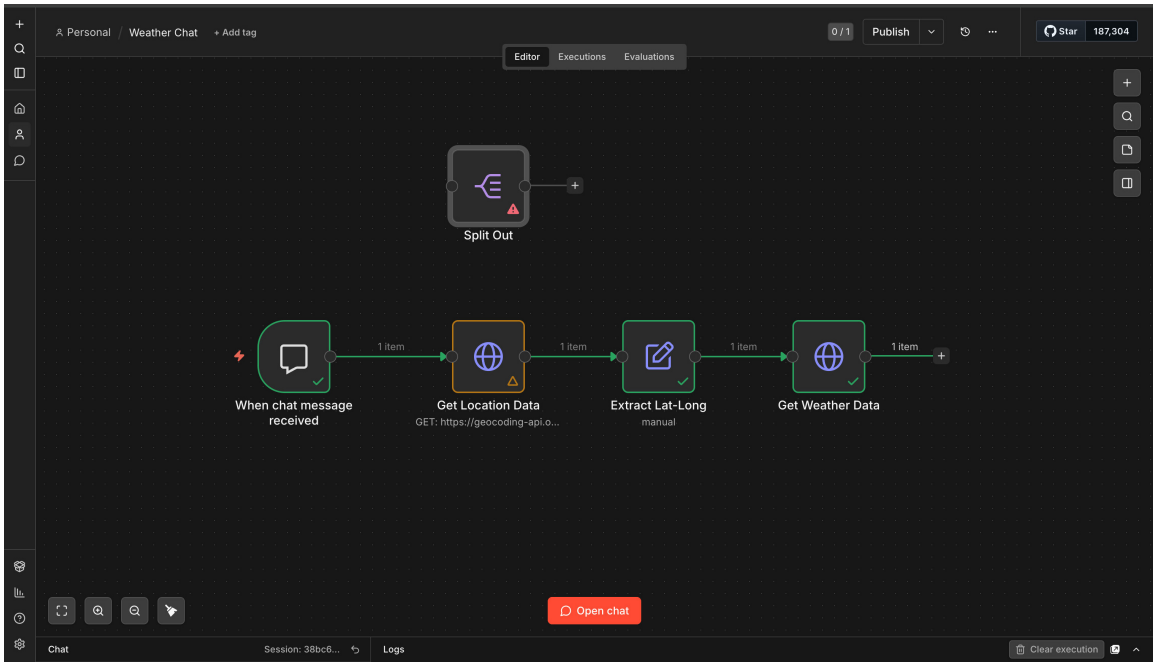
ဒီ Chat ကနေ စာပို့လိုက်ရင် Workflow ကို စသွားပြီး နောက်ထပ် Chat Message ပြန်အဝင်ကို စောင့်ဖို့ ပြောလိုက်တာပါ။ ပြီးသွားတဲ့အခါ ပိတ်လိုက်ပြီး၊ ဒီ Chat Trigger Node ကို ဟိုးရှေ့ဆုံးက မူလရှိနေတဲ့ **HTTP Request Node** နဲ့ ချိတ်ပေးလိုက်ပါ။

ဒီထိရပြီဆိုရင် ပုံမှာပြထားသလို **Open Chat** ခလုတ်ကိုနှိပ်ပြီးတော့ဘဲ ဖြစ်ဖြစ် Chat Trigger ကို Run ပြီးတော့ပဲဖြစ်ဖြစ် စာပို့ကြည့်လို့ရပါပြီ။



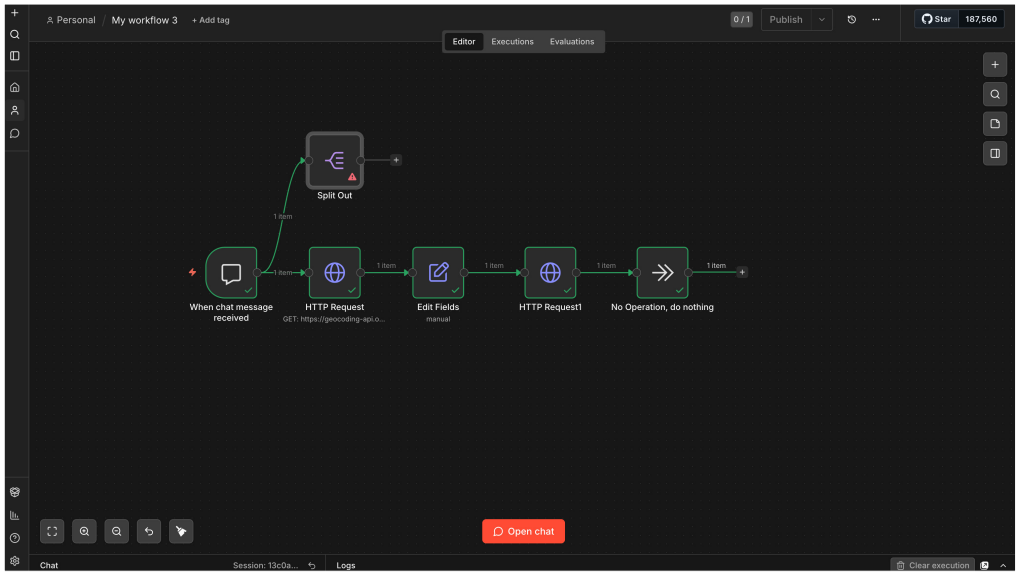
နမူနာပုံမှာ မြင်တွေ့ရသလို Warning Message တစ်ခုတွေ့ရရင် ပြဿနာမရှိပါဘူး။ နောက်ထပ် Chat Node တွေက ပြန်လာမယ့် Message ကို စောင့်ခိုင်းထားပေမဲ့ နောက်ထပ် Chat Node တွေ မထည့်ရသေးလို့ပါ။ နောက်တစ်ဆင့်အနေနဲ့ **Split Out** လို့

ခေါ်တဲ့ Node တစ်ခုကို ညာဘက်ခြမ်းက **[+]** လုပ်နှိပ်ပြီး ရှာထည့်လိုက်ပါ။ အခုလို ပုံစံ ဖြစ်ရပါမယ်။

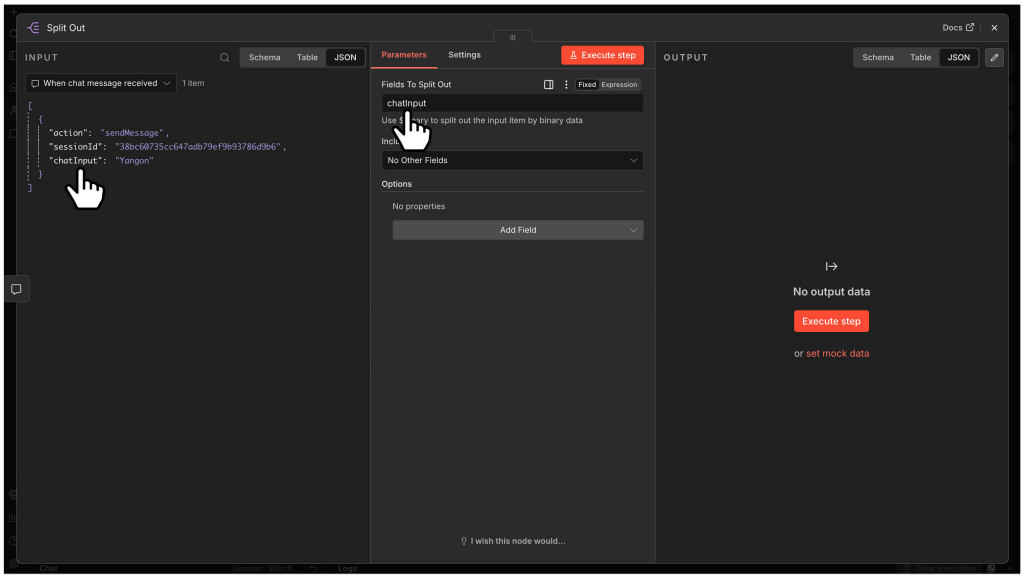


Split Out Node တစ်ခု လက်ရှိ Workflow နဲ့ မချိတ်ရသေးဘဲ သီးခြားရှိနေပါပြီ။ Node တွေကို လုပ်မဲ့အလုပ်ရဲ့ အဓိပ္ပာယ်ပေါ်လွင်အောင် အမည်လေးတွေ ပြောင်းပေးထားတာ သတိပြုပါ။ အဲ့ဒီလို သင့်တော်ရာအမည်လေးတွေ ပေးထားမှ Node တွေများလာတဲ့အခါ ကြည့်ရလွယ်မှာပါ။

ဆက်လက်ပြီး Chat Trigger ကို Split Out နဲ့လည်း ဆွဲချိတ်လိုက်ပါ။ ဒါကြောင့် Chat Trigger က **HTTP Node** နဲ့ရော **Split Node** နဲ့ပါ နှစ်ခုချိတ်ထားတာ ဖြစ်သွားပါလိမ့်မယ်။



Split Out Node ကို Double-Click နှိပ်ပြီးဖွင့်လိုက်ပါ။

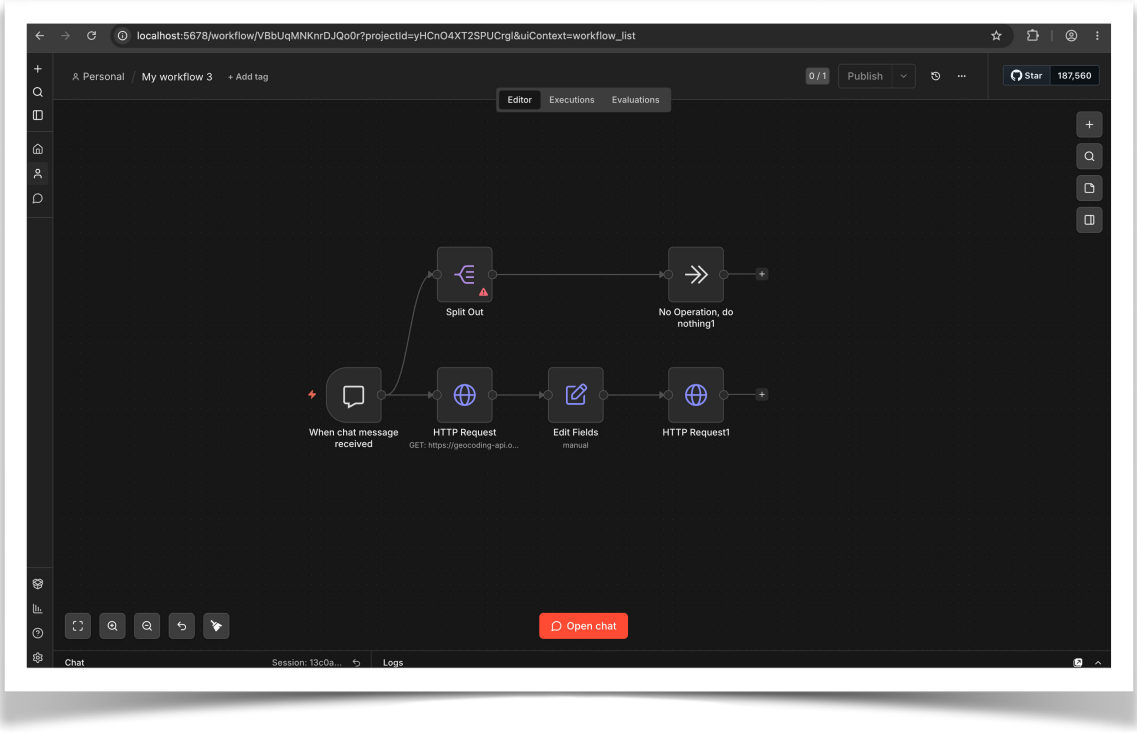


Fields to Split Out မှာ **chatInput** လို့ ထည့်ပေးလိုက်ပါ။ ဒီထိရသွားရင် Chat Message ပို့လိုက်တဲ့အခါ Message က **HTTP Node** ဆီကိုလည်း ရောက်သွားမယ်၊ **Split Node** ဆီကိုလည်း ရောက်သွားပါမယ်။

ပုံမှန်အားဖြင့် Split Out Node ကို အဝင်ဒေတာတစ်ခုကနေ အထွက်ဒေတာတွေ အများကြီးခွဲထုတ်ဖို့ သုံးရပါတယ်။ ဒီနမူနာမှာ အထွက်ဒေတာတွေ အများကြီးခွဲထုတ် မထားပါဘူး။ အဝင်လည်း တစ်ခုပဲရှိသလို အထွက်လည်း တစ်ခုပဲရှိပါတယ်။ ဒါကြောင့် ဒီနေရာမှာ ဒီ Split Out Node ကို မသုံးလည်း ရပါတယ်။

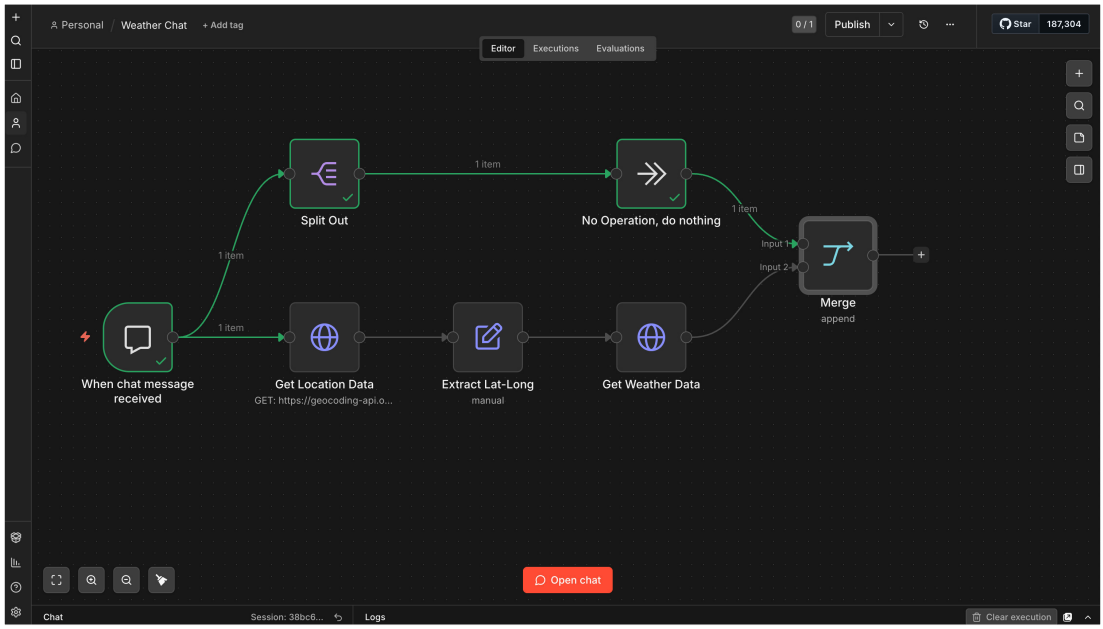
ကျွန်တော်စာရေးသူရဲ့ မူလအိုင်ဒီယာက တစ်မျိုးဖြစ်ပြီး လုပ်ရင်းနဲ့ စိတ်ကူးပြောင်းသွားပေမဲ့ သုံးလက်စ Node ကို မပြောင်းတော့ဘဲ ဆက်သုံးလိုက်တာပါ။

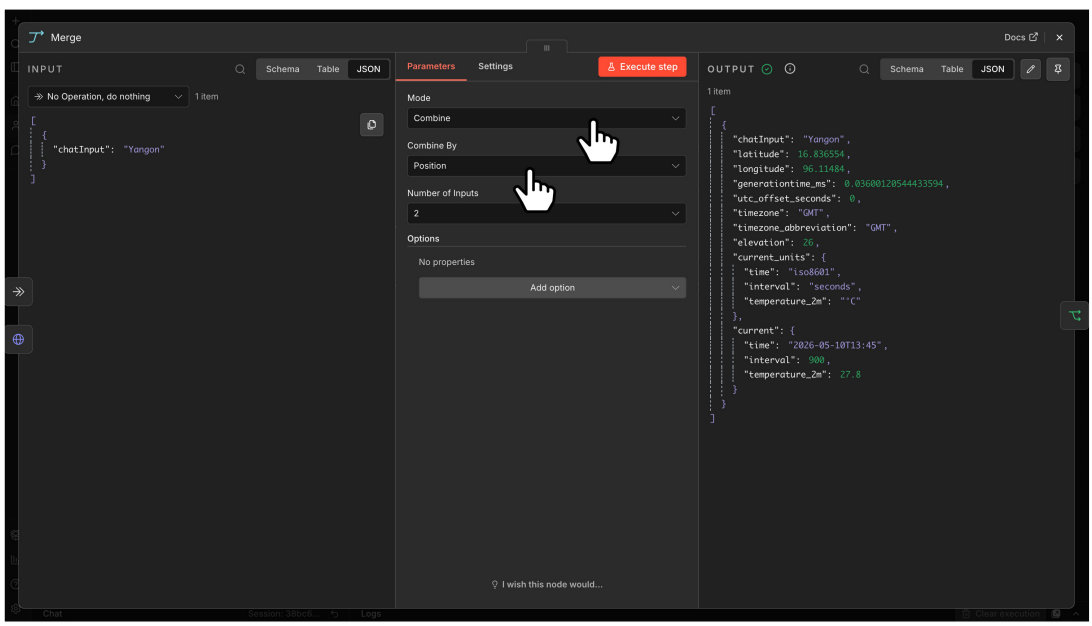
ဆက်လက်ပြီး မူလရှိနေတဲ့ Do Nothing Node ကို ဖျက်လိုက်ပါ။ Split Out Node နောက်က **[+]** ကို နှိပ်ပြီး **Do Nothing Node** နောက်တစ်ခု ပြန်ထည့်လိုက်ပါ။ အခုလိုပုံစံ ဖြစ်သွားရပါမယ်။



Chat Message က လမ်းကြောင်းနှစ်ခုနဲ့ သွားတာ ဖြစ်သွားပါပြီ။ တစ်ခုက HTTP Node တွေကို ဖြတ်သွားပြီး နောက်တစ်ခုက Split နဲ့ Nothing ကို ဖြတ်သွားတာပါ။

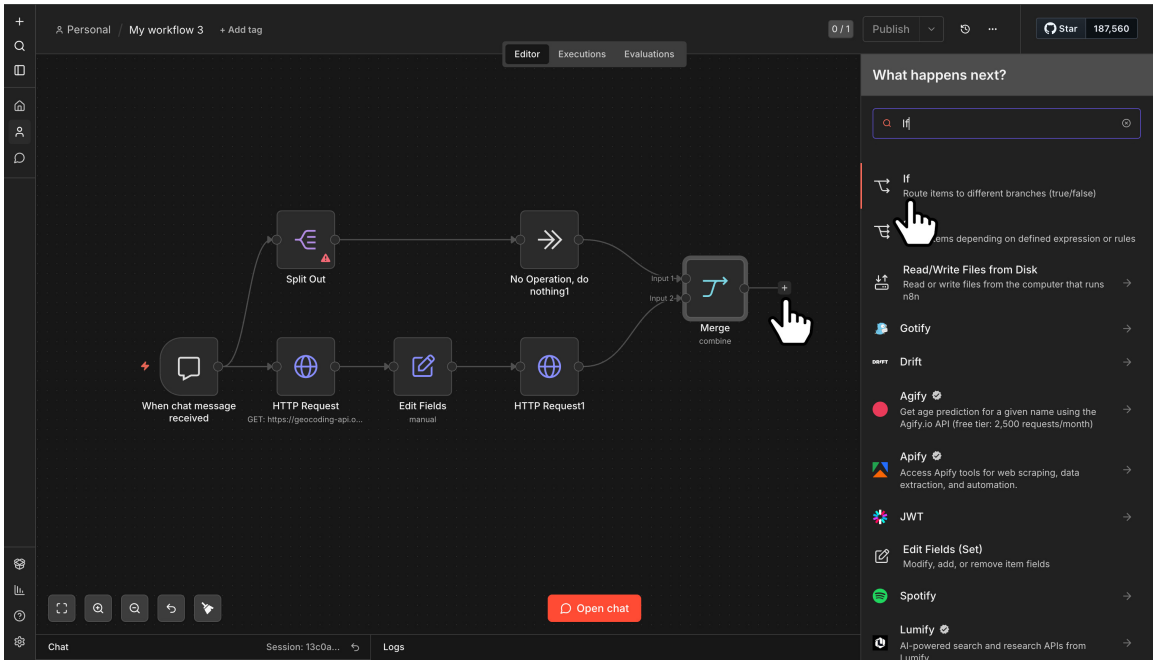
နောက်တစ်ဆင့်မှာ **Merge Node** ကို ရှာထည့်လိုက်ပါ။ Merge Node က အဝင်တွေ အများကြီးကို တစ်ခုတည်းဖြစ်အောင် ပေါင်းပေးပါတယ်။ အခုလိုပုံစံဖြစ်ရပါမယ်။


Personal / Weather Chat + Add tag
0 / 1
Publish
Star 187,304
Editor
Executions
Evaluations
Split Out
1 item
No Operation, do nothing
1 item
Input 1
Input 2
Merge
append
1 item
1 item
When chat message received
Get Location Data
GET: https://geocoding-api.o...
Extract Lat-Long
manual
Get Weather Data
Open chat
Chat
Session: 38bc6...
Logs
Clear execution


Merge
Docs
INPUT
Schema
Table
JSON
No Operation, do nothing
1 item
{
"chatInput": "Yangon"
}
Parameters
Settings
Execute step
Mode
Combine
Combine By
Position
Number of Inputs
2
Options
No properties
Add option
I wish this node would...
OUTPUT
Schema
Table
JSON
1 item
{
"chatInput": "Yangon",
"latitude": 16.836554,
"longitude": 96.11484,
"generationtime_ms": 0.03600120544433594,
"utc_offset_seconds": 0,
"timezone": "GMT",
"timezone_abbreviation": "GMT",
"elevation": 26,
"current_units": {
"time": "iso8601",
"interval": "seconds",
"temperature_2m": "°C"
},
"current": {
"time": "2026-05-10T13:45",
"interval": 900,
"temperature_2m": 27.8
}
}

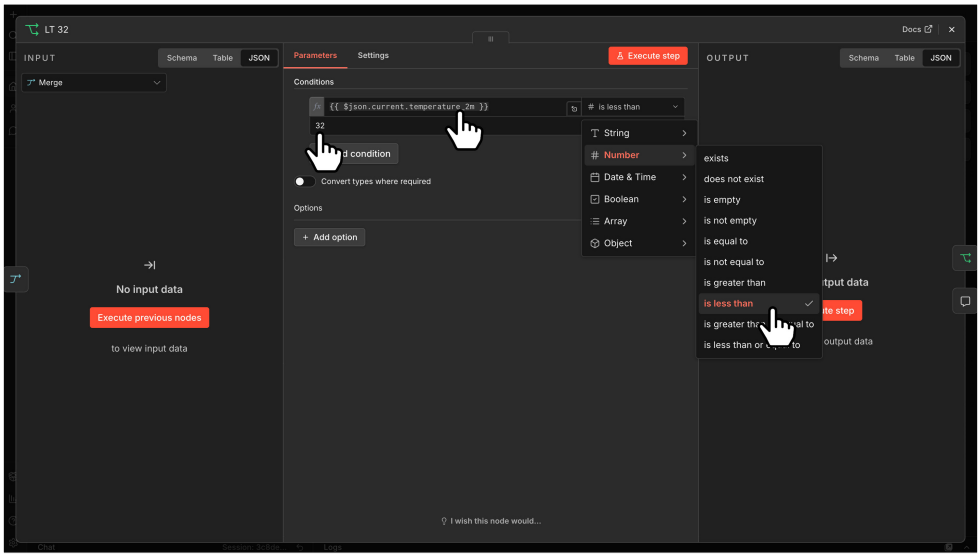
လမ်းကြောင်းနှစ်ခုကို တစ်ခုတည်းဖြစ်အောင် ပြန်ပေါင်းလိုက်ပါပြီ။ Merge Node ကို Double-Click နှိပ်ဖွင့်ပြီး Mode ကို Combine နဲ့ Combine By ကို Position ရွေးပေးပါ။

နမူနာပုံမှာ ညာဘက်ခြမ်းက Output ကို ကြည့်လိုက်ရင် Chat Message နဲ့ Weather ဒေတာကို ပေါင်းစပ်ပေးထားတာ တွေ့ရမှာဖြစ်ပါတယ်။ နောက်တစ်ဆင့်မှာ Merge ရဲ့နောက်က [+] ခလုတ်နဲ့ **If Node** ကို ရှာထည့်လိုက်ပါ။

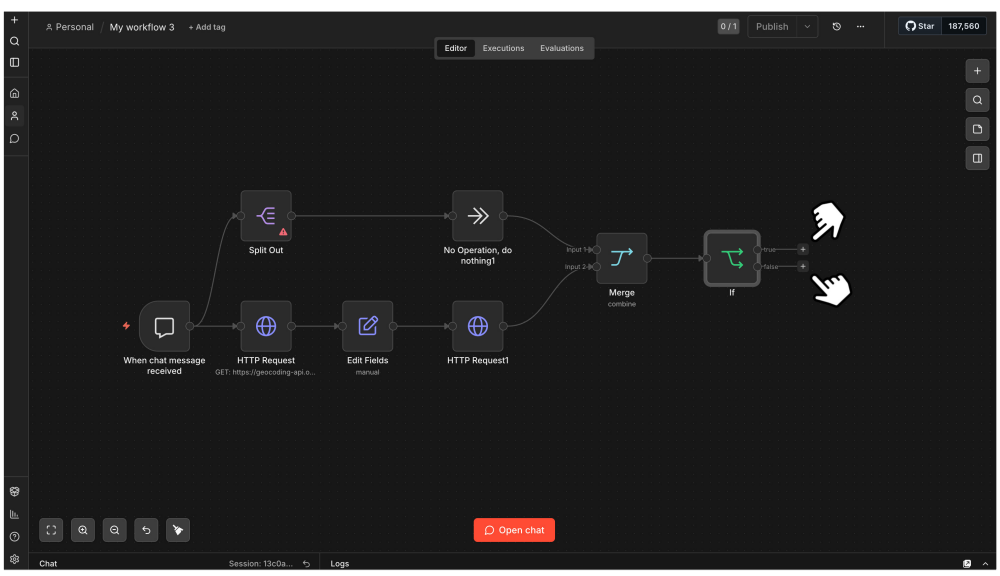


If Node ကို ဒေတာရဲ့ အခြေအနေ စစ်ဖို့သုံးပါတယ်။ နမူနာမှာ ပြထားသလို Conditions နေရာမှာ `{{ $json.current.temperature_2m }}` လို့ ထည့်ပေးလိုက်ပါ။ Input ကနေရတဲ့ ရာသီဥတုအပူချိန်ကို ယူလိုက်တာပါ။

တကယ်တော့ ကိုယ့်ဘာသာ ထည့်စရာမလိုပါဘူး။ ဘယ်ဘက်ခြမ်းမှာ Input ဒေတာရှိနေရင် ဆွဲထည့်လိုက်လို့ရပါတယ်။ နမူနာမှာ မရှိသေးလို့ ကိုယ့်ဘာသာ ထည့်ပြလိုက်တာပါ။



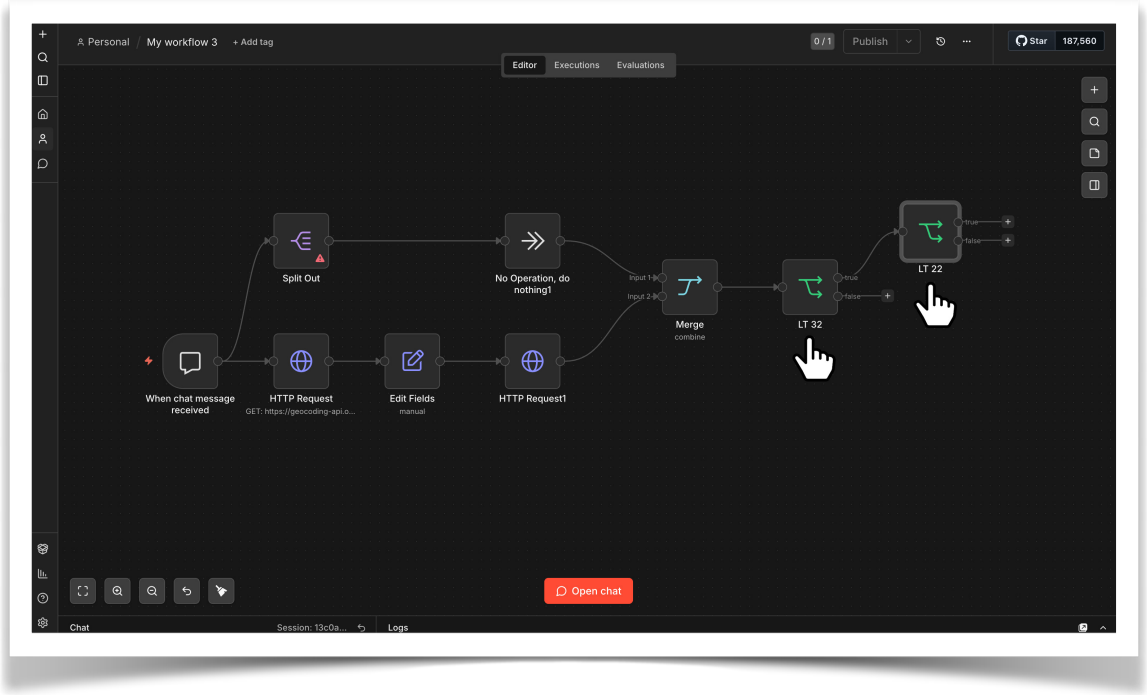
ပုံမှာပြထားသလို **Number → Is Less Than** ကို ရွေးပြီး **32** လို့ ထည့်ပေးလိုက်ပါ။ Temperature က 32 ဒီဂရီအောက်လားလို့ စစ်လိုက်တာပါ။



If Node ရဲ့ နောက်မှာ **true** နဲ့ **false** ဆိုပြီး လမ်းကြောင်းအခွဲ နှစ်ခုရှိနေတာကို တွေ့ရမှာပါ။ စစ်ထားတဲ့ Condition အရ အပူချိန် (၃၂) မကျော်ရင် **true** နောက်ကို လိုက်သွားမှာဖြစ်ပြီး၊ ကျော်ရင် **false** နောက်ကို လိုက်သွားမှာပါ။

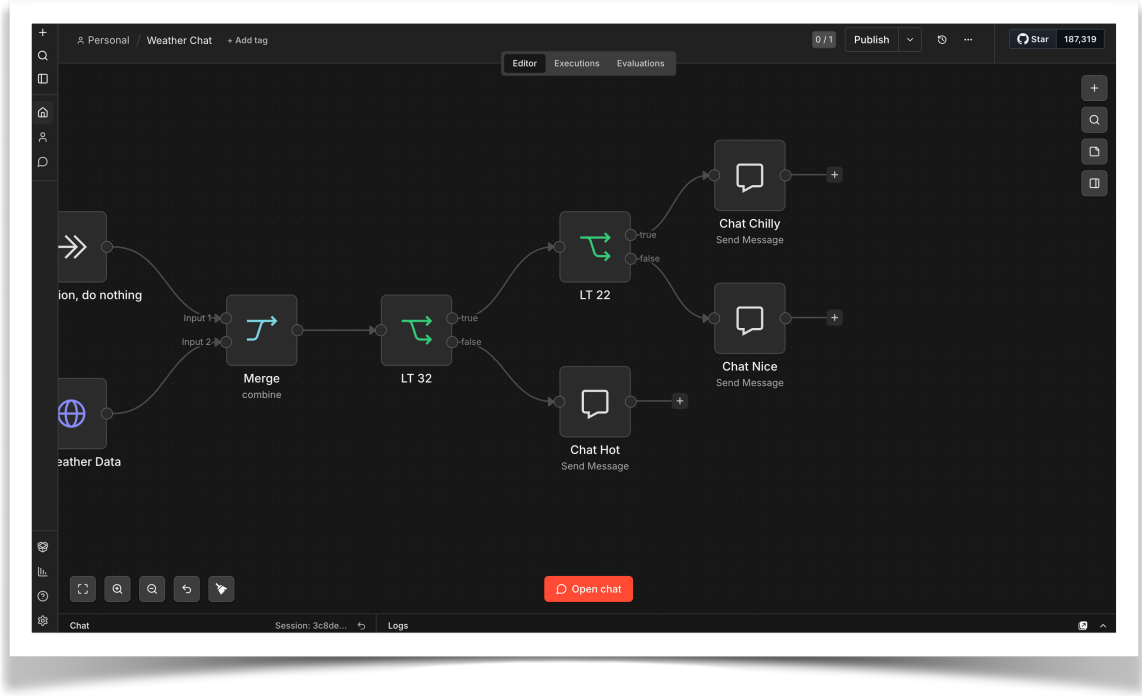
true ရဲ့ နောက်က **[+]** ကိုနှိပ်ပြီး နောက်ထပ် **If Node** တစ်ခုထပ်ထည့်လိုက်ပါ။

ဒီတစ်ခါလည်း Condition ကို `{{ $json.current.temperature_2m }}` လို့ပဲ ပေးပြီး **Number → Is Less Than** ကိုရွေးပြီး **22** လို့ပေးလိုက်ပါ။ စုံသွားရင် ရလဒ်က အခုလို ဖြစ်ရပါမယ်။



If Node နှစ်ဆင့် ဖြစ်သွားတာပါ။ ပထမတစ်ဆင့်မှာ Temperature က 32 အောက်လား စစ်ပါတယ်။ ဟုတ်တယ်ဆိုရင် နောက်တစ်ဆင့်မှာ Temperature က 22 အောက်လားလို့

ထပ်ဆင့် စစ်ပါတယ်။ ဒါကို ထင်ရှားအောင် Node တွေရဲ့ အမည်ကို **LT 32** နဲ့ **LT 22** လို့ ပေးထားပါတယ်။ နောက်ထပ်တစ်ဆင့်အနေနဲ့ အခုလို **Chat Node** လေးတွေရှာပြီး တွဲ ပေးလိုက်ပါ။

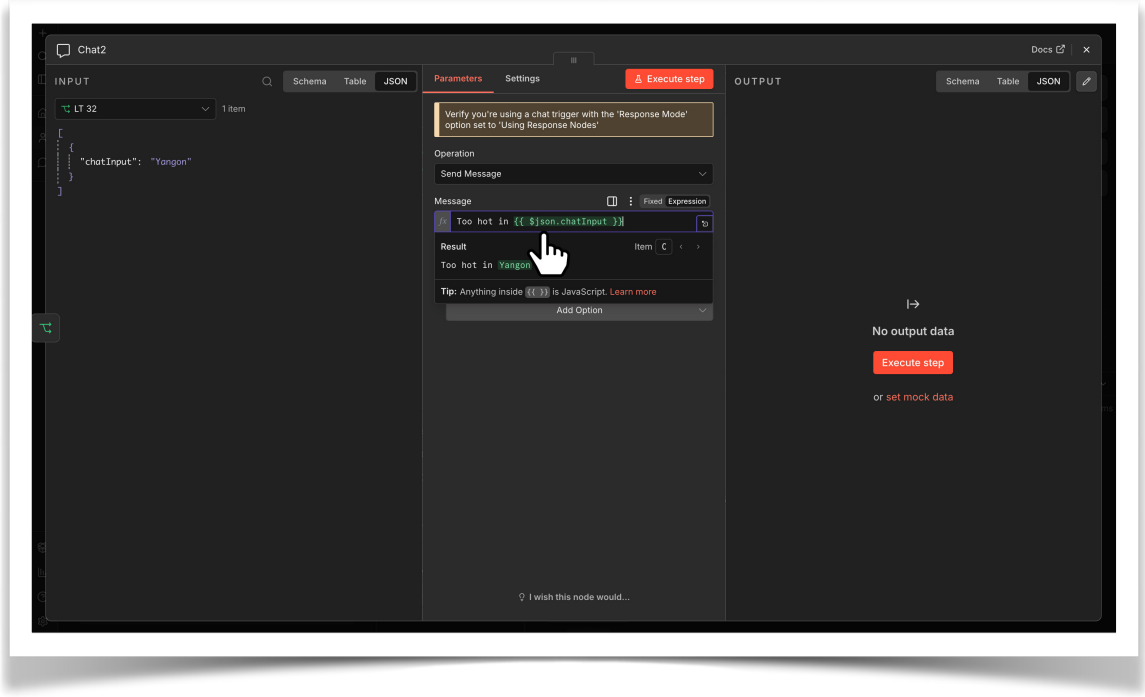


Chat Node လေးတွေကို အမည်လေးတွေလည်း ပေးထားပါတယ်။ အပူချိန် (၃၂) ကျော် နေရင် **Chat Hot** ကို ရောက်ပါမယ်။ အပူချိန် (၂၂) အောက်ဆိုရင် **Chat Chilly** ကို ရောက်ပါမယ်။ အပူချိန် (၃၂) ကျော်ပေမဲ့ (၂၂) မကျော်ရင် **Chat Nice** ကို ရောက်မှာပါ။

Programming တွေဘာတွေ လေ့လာဖူးသူဆိုရင် ဒီလို true/false Logical Condition တွေ ကို အလွယ်တကူ နားလည်ကြပါလိမ့်မယ်။ မသင်ဖူးရင်လည်း ကိစ္စမရှိပါဘူး။ ဘယ်လို

အခြေအနေမှာ ဘယ်လမ်းကြောင်းနောက်ကို လိုက်သွားမလဲဆိုတာ မျက်မြင်ပဲ သေချာလိုက်ကြည့်လိုက်။

Chat Hot Node မှာ အခုလို ထည့်ပေးပါ။

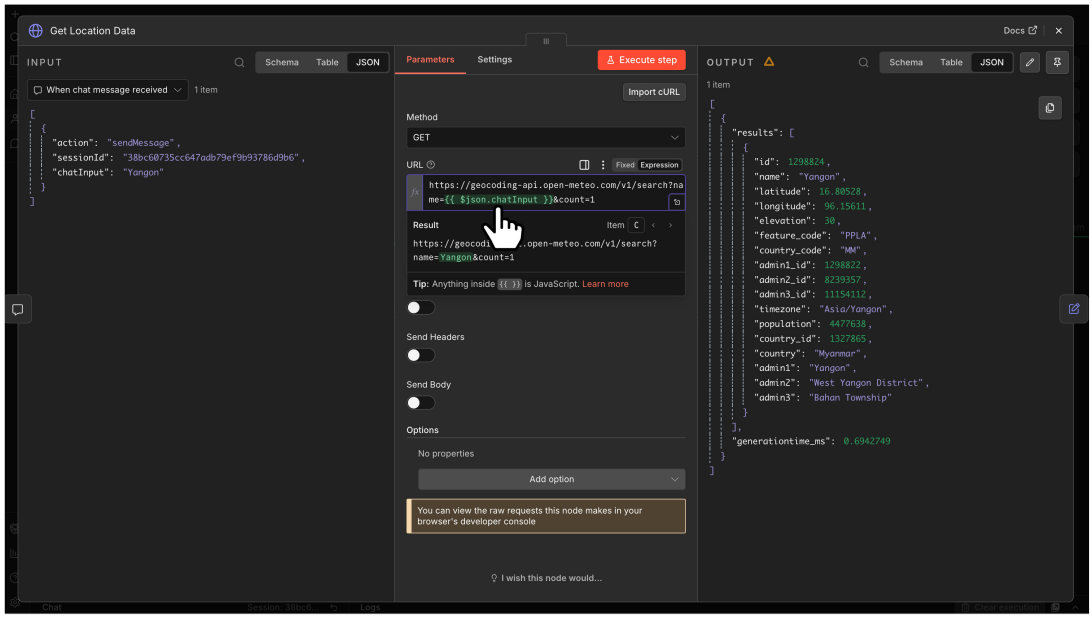


Message နေရာမှာ `Too hot in {{ $json.chatInput }}` လို့ ထည့်ပေးလိုက်တာပါ။ သီးခြား လမ်းကြောင်းတစ်ခုနဲ့ ဟိုးရှေ့ဆုံးက **chatInput** ကို သယ်ဆောင်လာတာ ဒီနေရာမှာ အသုံးလိုမှာ မို့လို့ပါ။

အလားတူပဲ **Chat Chilly** ရဲ့ Message မှာ `Chilly in {{ $json.chatInput }}` လို့ ထည့်ပြီး **Chat Nice** ရဲ့ Message မှာ `Nice weather in {{ $json.chatInput }}` တို့ကို ထည့်ပေးလိုက်ပါ။

စမ်းကြည့်လို့ရပါပြီ။ စစခြင်း **Chat Trigger** မှာ Message တစ်ခုခု ရိုက်ထည့်လိုက်ရင် တစ်ဆင့်ပြီး တစ်ဆင့် အလုပ်လုပ်သွားပြီး နောက်ဆုံးမှာ ရာသီဥတုပူရင် ရန်ကုန်မှာ ပူတယ်၊ အေးရင် ရန်ကုန်မှာ အေးတယ်၊ စသည်ဖြင့် အကြောင်းပြန်တဲ့ Chat Message ကို ပြန်ရတာ တွေ့ရမှာ ဖြစ်ပါတယ်။

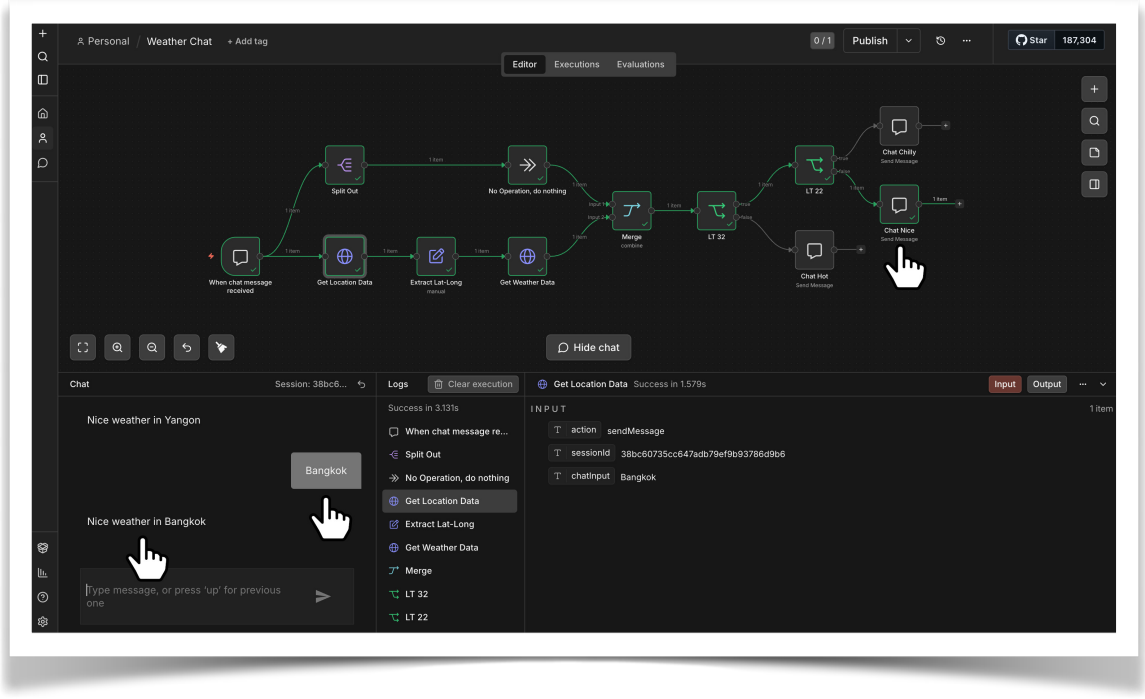
တစ်ခုရှိတာက Location Data ယူတဲ့ **HTTP Node** မှာ **Yangon** ရဲ့ ဒေတာကို ယူထားလို့ ရန်ကုန်ရဲ့ ဒေတာကိုပဲ အမြဲပြနေမှာပါ။ အဲ့ဒီလို မဖြစ်အောင် Location Data ယူတဲ့ HTTP Node ကို အခုလို ပြင်ပေးပါ။



မူလ **name=Yangon** နေရာမှာ `name={{ $json.chatMessage }}` လို့ အစားထိုးပေးလိုက်တာပါ။ ဒါကြောင့် တည်နေရာအချက်အလက် ယူတဲ့အခါ Yangon ရဲ့

တည်နေရာကို ပုံသေမယူတော့ဘဲ Chat Message မှာ ရိုက်ထည့်လိုက်တဲ့ ဒေသရဲ့ တည်နေရာအချက်အလက်ကို ယူပေးသွားမှာပဲ ဖြစ်ပါတယ်။

အားလုံးပြည့်စုံသွားပါပြီ။ စမ်းကြည့်လို့ရပါပြီ။



နမူနာမှာ Chat ကိုဖွင့်ပြီး **Bangkok** လို့ရိုက်ထည့်လိုက်တဲ့အခါ၊ စမ်းသပ်ချိန်မှာ ဘန်ကောက်မြို့အပူချိန် (၃၂) ထက်နည်းပြီး (၂၂) ကို မကျော်တဲ့အတွက် **Nice weather in Bangkok** ဆိုတဲ့ အကြောင်းပြန်စာကို ရရှိတာ တွေ့ရမှာ ဖြစ်ပါတယ်။

ဒါဟာ AI Agent တွေဘာတွေ မပါသေးဘဲ၊ Chat လုပ်လို့ရတဲ့ Workflow တစ်ခုကို ရသွားတာပါ။ AI Agent အကြောင်းကို နောက်သင့်တော်တဲ့အခန်း ရောက်မှာ ဆက်လေ့လာကြပါမယ်။

အခန်း (၇) - Invoice & Reminder Workflow

ဒီအခန်းမှာ လက်တွေ့ကျတဲ့ လုပ်ငန်းသုံး Workflow လေးတစ်ခုလောက် လုပ်ကြည့်ကြပါမယ်။ Invoice Workflow ခေါ်ငွေတောင်းခံလွှာ ပေးပို့တဲ့ Workflow လုပ်ကြည့်ကြမှာပါ။

လုပ်ငန်းတွေမှာ ကုန်ပစ္စည်းရောင်းချတာပဲဖြစ်ဖြစ်၊ ဝန်ဆောင်မှုပေးတာပဲ ကျသင့်ငွေကို Customer ဆီက တောင်းခံတဲ့အခါ Invoice တွေ ပေးပို့တောင်းခံကြလေ့ ရှိပါတယ်။ တချို့လည်း ကြိုတင်ပေးချေရတယ်။ တချို့လည်း လုပ်ငန်းပြီးဆုံးတော့မှ ပေးချေရတယ်။ တချို့လည်း အရစ်ကျ ပေးချေရတယ်။ ပုံစံအမျိုးမျိုးနဲ့ ဖြစ်နိုင်ပါတယ်။

Acme Co., Ltd. ဆိုတဲ့လုပ်ငန်းတစ်ခု ရှိတယ်ဆိုကြပါစို့။ ဒီလုပ်ငန်းမှာ Customer ဆီက ငွေတောင်းခံဖို့ ရှိတဲ့အခါ...

> ၁။ သက်ဆိုင်ရာဌာနက Invoice ကို PDF ဖိုင်အနေနဲ့ Google Drive ထဲမှာ ထည့်လိုက်ပါတယ်။
>
> ၂။ တာဝန်ရှိသူမန်နေဂျာက Invoice ကို စစ်ဆေးပြီး ပြည့်စုံမှန်ကန်တယ်ဆိုရင် ငွေတောင်းခံရန်စာရင်း Google Sheets ထဲမှာ Record လုပ်လိုက်ပါတယ်။

၃။ ဝန်ထမ်းတစ်ဦးက မနက်ရုံးရောက်တဲ့အခါ ငွေတောင်းခံရန် စာရင်းထဲမှာ ရှိနေတဲ့ Customer တွေထံ ငွေတောင်းခံစာကို Gmail အသုံးပြုပြီး၊ Email နဲ့ ပေးပို့ပါတယ်။

၄။ ပေးပို့ပြီးတဲ့အခါ ငွေတောင်းခံလွှာ ပေးပို့ပြီးကြောင်း Google Sheets မှာ စာရင်းသွင်းလိုက်ပါတယ်။

၅။ ငွေလက်ခံရရှိတဲ့အခါ သက်ဆိုင်ရာဌာနက လက်ခံရရှိပြီးဖြစ်ကြောင်း Google Sheets မှာ Update လုပ်လိုက်ပါတယ်။

၆။ ဝန်ထမ်းတစ်ဦးက မနက်ရုံးရောက်တဲ့အခါ ငွေတောင်းခံလွှာပေးပို့ထားပေမဲ့ (၃) ရက်အတွင်း ငွေပေးချေခြင်းမရှိသေးသူတွေထံ Email နဲ့ Reminder ပေးပို့ပါတယ်။

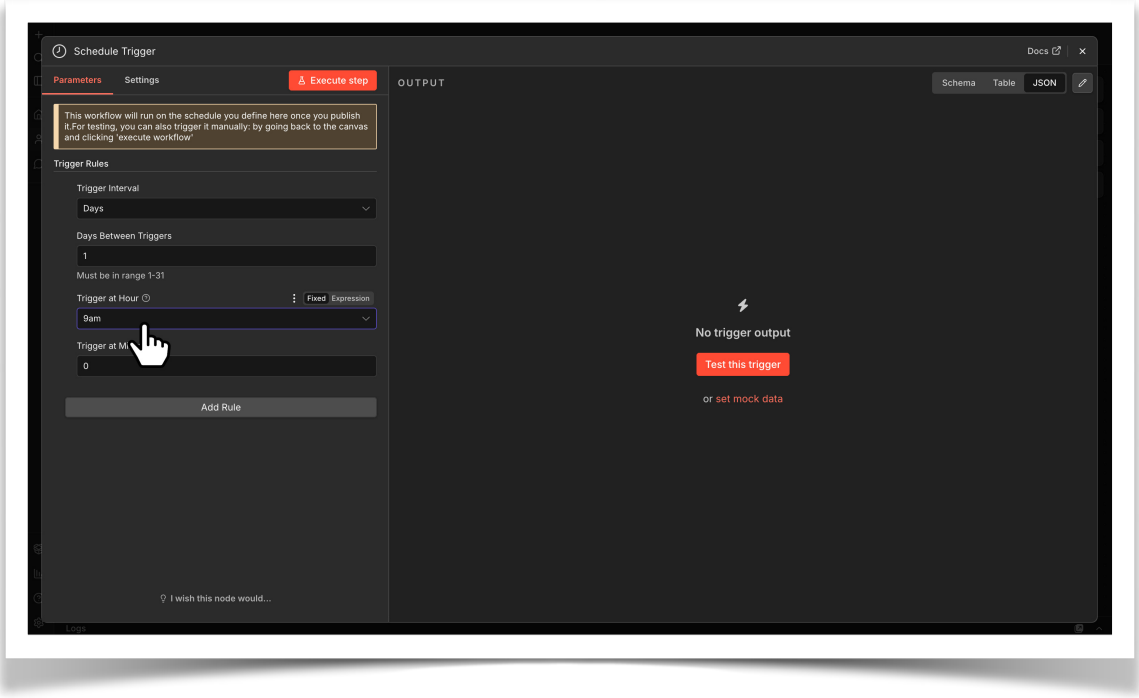
၇။ ပေးပို့ပြီးတဲ့အခါ Reminder ဘယ်နှစ်ကြိမ် ပို့ပြီးပြီလဲ Google Sheets မှာ မှတ်တမ်းသွင်းလိုက်ပါတယ်။

လုပ်ရတဲ့အဆင့်တွေ တော်တော်များပါတယ်။ ဒီအဆင့်အားလုံးကို Automate လုပ်ထားလို့ရပေမဲ့ ဒီနေရာမှာ နမူနာအနေနဲ့ **အဆင့် (၃)** ကနေစပြီး လုပ်ကြည့်ကြမှာပါ။ Google Drive ထဲမှာ Invoice ဖိုင်တွေရှိနေပြီး Google Sheets ထဲမှာ ငွေတောင်းခံရန်စာရင်း ရှိနေတယ်လို့ သဘောထားရမှာ ဖြစ်ပါတယ်။

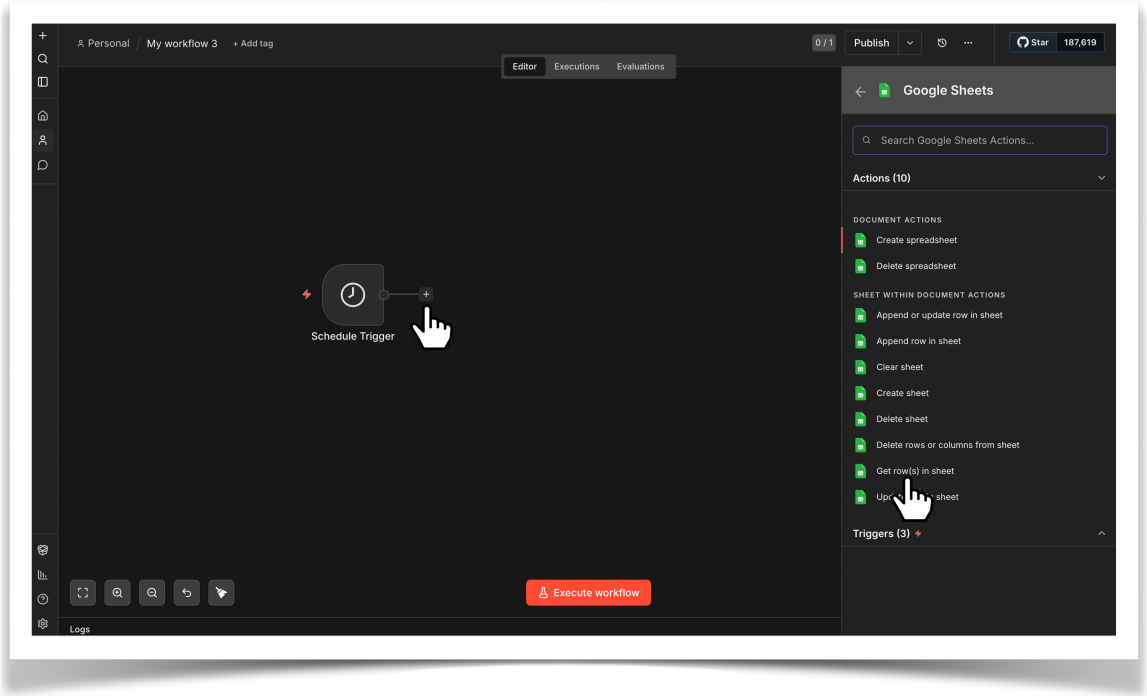
စလိုက်ကြရအောင်...

ဘယ်ဘက်အပေါ်ထောင့်က **[+]** ခလုတ်ကိုနှိပ်ပြီး Workflow အသစ်တစ်ခုတည်ဆောက်လိုက်ပါ။ Node တွေ ဘယ်လိုထည့်ရလဲ၊ ဘယ်လိုချိတ်ရလဲ သိနေပြီမို့လို့ တချို့အဆင့်တွေကို ပုံတွေနဲ့ အကုန် မပြတော့ပါဘူး။ လိုအပ်သလောက်ပဲ ပြသွားပါမယ်။

ပထမဆုံးအနေနဲ့ **Schedule Trigger Node** တစ်ခုကနေ စထည့်လိုက်ပါ။



ပုံမှာပြထားသလို **Trigger at Hour** မှာ 9am ကို ရွေးလိုက်ပါ။ နေ့စဉ် မနက် (၉) နာရီ ရောက်တိုင်း အလုပ်လုပ်မယ်ဆိုတဲ့အဓိပ္ပာယ်ပါ။ နောက်တစ်ဆင့်မှာ **Google Sheets** ကို ရှာပြီး **Get row(s) in sheets Node** ကို ထည့်လိုက်ပါ။

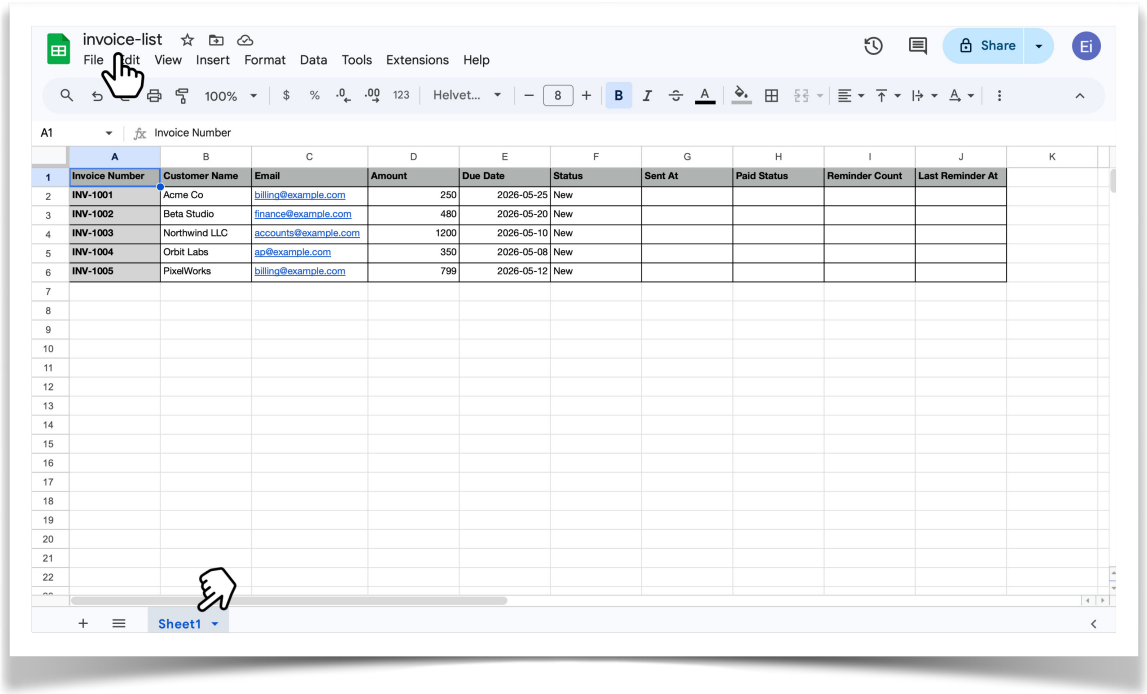


ဒီအဆင့်ကိုရောက်တဲ့အခါ Workflow ကို ဆက်မလုပ်သေးဘဲ လိုအပ်တဲ့ Google Sheets နမူနာတစ်ခုလောက်ကို အရင်လုပ်ဖို့လိုလာပါတယ်။ ဒီလင့်ကိုသွားပြီး လိုအပ်ရင် Google Account နဲ့ Login ဝင်ထားလိုက်ပါ။

https://docs.google.com/spreadsheets/create

ပြီးတဲ့အခါ နမူနာ ဒေတာလေးတချို့ကို အခုလိုထည့်ပေးလိုက်ပါ။ အကုန် မထည့်ချင်ရင်တောင် စမ်းစရာ တစ်ခုနှစ်ခုလောက် ထည့်ထားရင် ရပါတယ်။ အပေါ်ဆုံးက Header Row မှာပါတဲ့ Column တွေ မှန်အောင်ထည့်ပြီး Email နေရာမှာ ကိုယ်စမ်းသပ်ပေးပို့လိုတဲ့ Email အမှန်ထည့်ပေးပါ။

ဖိုင်အမည်ကို **invoice-list** လို့ ပေးထားပြီး **Sheet1** မှာပဲ စာရင်းကို ထည့်ထားပါတယ်။



Invoice Number	Customer Name	Email	Amount	Due Date	Status	Sent At	Paid Status	Reminder Count	Last Reminder At
INV-1001	Acme Co	billing@example.com	250	2026-05-25	New				
INV-1002	Beta Studio	finance@example.com	480	2026-05-20	New				
INV-1003	Northwind LLC	accounts@example.com	1200	2026-05-10	New				
INV-1004	Orbit Labs	ap@example.com	350	2026-05-08	New				
INV-1005	PixelWorks	billing@example.com	799	2026-05-12	New				

ပါရမယ့် Header Column တွေက ဒီလိုပါ။

- Invoice Number
- Customer Name
- Email
- Amount
- Due Date
- Status
- Sent At
- Paid Status
- Reminder Count

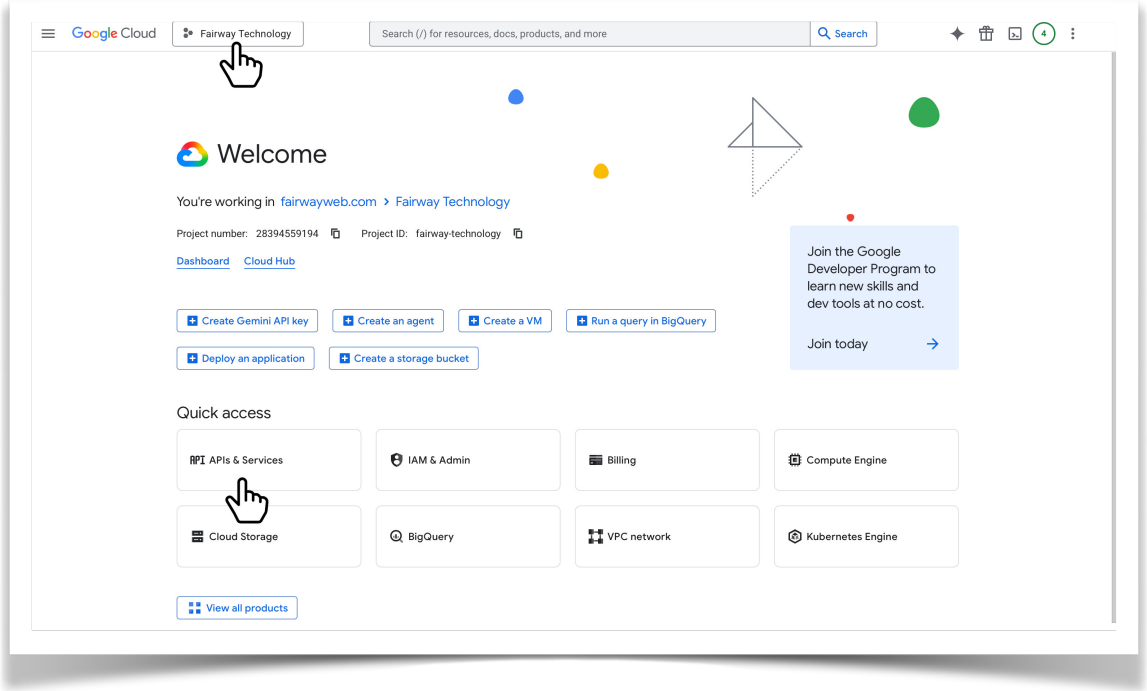
- Last Reminder At

ပြီးတဲ့အခါ Google Sheets နဲ့ လိုအပ်တဲ့ Google ရဲ့လုပ်ဆောင်ချက်တွေကို n8n ကနေ ဆက်သွယ်အသုံးပြုခွင့် ရစေဖို့အတွက် **API Key** တွေ ယူတဲ့အလုပ်ကို လုပ်ရပါမယ်။ နည်းနည်း အလုပ်ရှုပ်ပါလိမ့်မယ်။

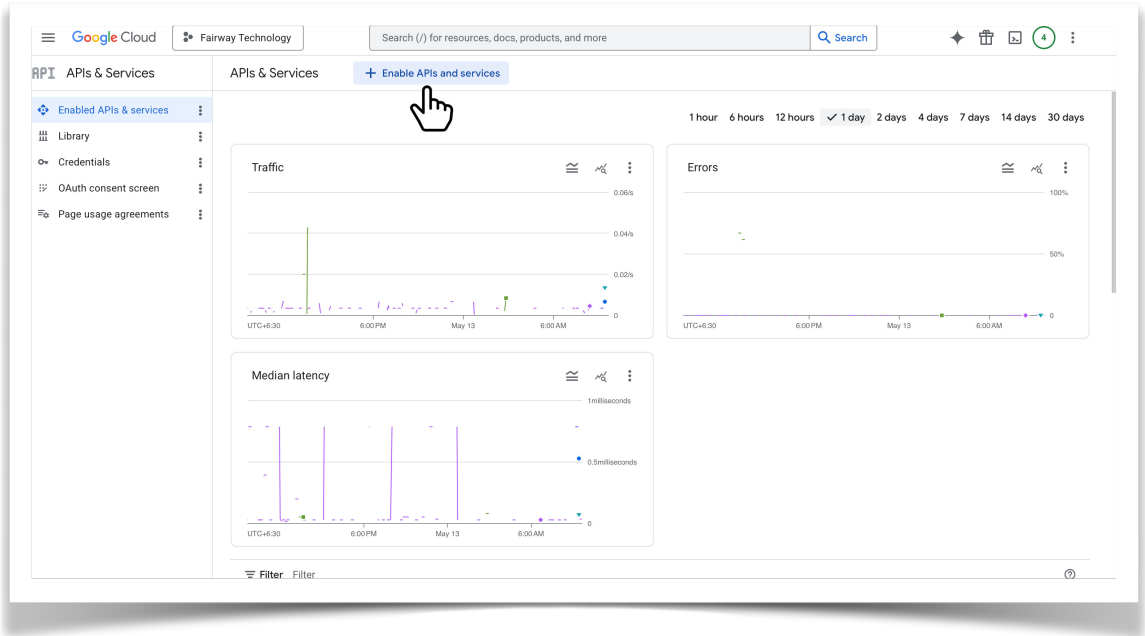
Google Cloud Console ကို ဒီလိပ်စာကနေ သွားလိုက်ပါ။

https://console.cloud.google.com/

Google အကောင့်ရှိဖို့နဲ့ ရှိတဲ့အကောင့်နဲ့ Login ဝင်ဖို့ကိစ္စကိုတော့ ကိုယ့်ဘာသာလုပ်ရပါလိမ့်မယ်။ အကောင့်ရှိပြီး Login ဝင်ပြီးရင် ဘာဆက်လုပ်ရမလဲ ပြောပြပါမယ်။



အထက်ကပုံမှာ ပြထားသလို Google Cloud တံဆိပ်နားက ခလုတ်ကိုနှိပ်ပြီး ပရောဂျက်သစ်တစ်ခု တည်ဆောက်ပါ။ ပြီးတဲ့အခါ **APIs & Services** ကို နှိပ်ပါ။



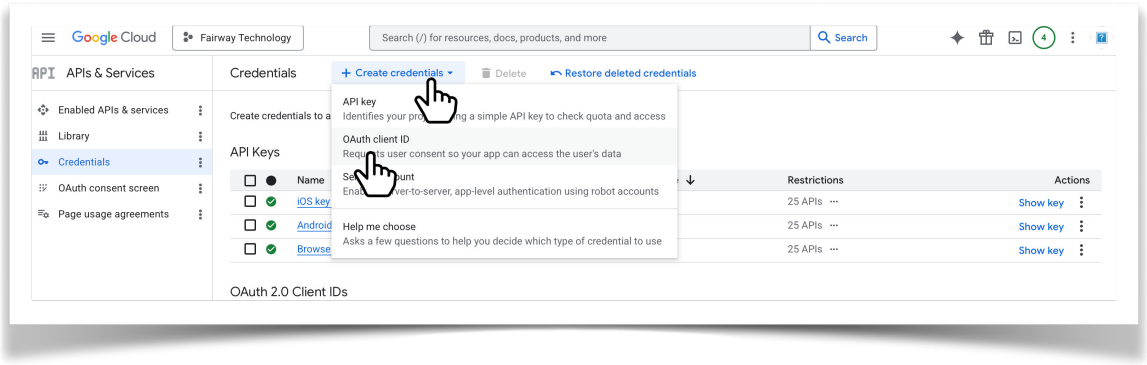
Enable APIs and services ကို နှိပ်ပါ။ **Google Sheets API** ကို ရှာပြီး **Enable** လုပ်လိုက်ပါ။ တစ်လက်စတည်း **Google Drive API**, **Gmail API** နဲ့ **YouTube Data API v3** တို့ကို ရှာပြီး အကုန်လုံး Enable လုပ်ထားလိုက်ပါ။

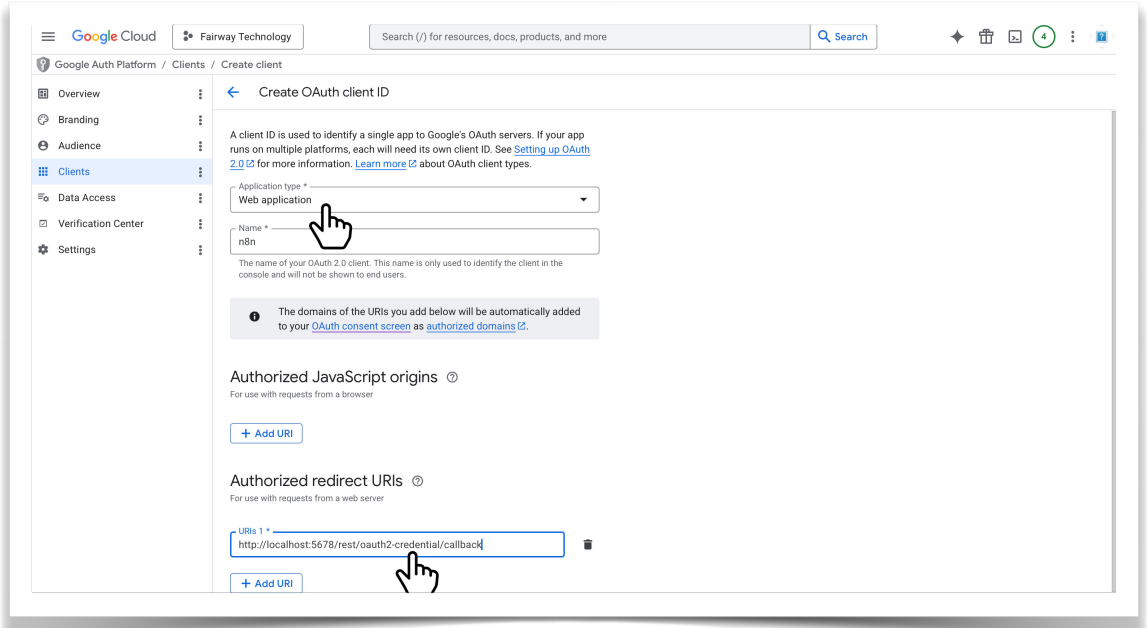
Enable လုပ်ရမယ့်စာရင်းကို ထပ်မံဖော်ပြလိုက်ပါတယ်။

- Google Sheets API
- Google Drive API
- Gmail API
- YouTube Data API v3

ဒါတွေအကုန် Enable လုပ်ထားမှ နောက်အဆင့်တွေအတွက် အသင့်ဖြစ်နေမှာပါ။

APIs and Services ကို ပြန်သွားလိုက်ပြီး **+ Create credential** ထဲက OAuth client ID ကို ရွေးလိုက်ပါ။

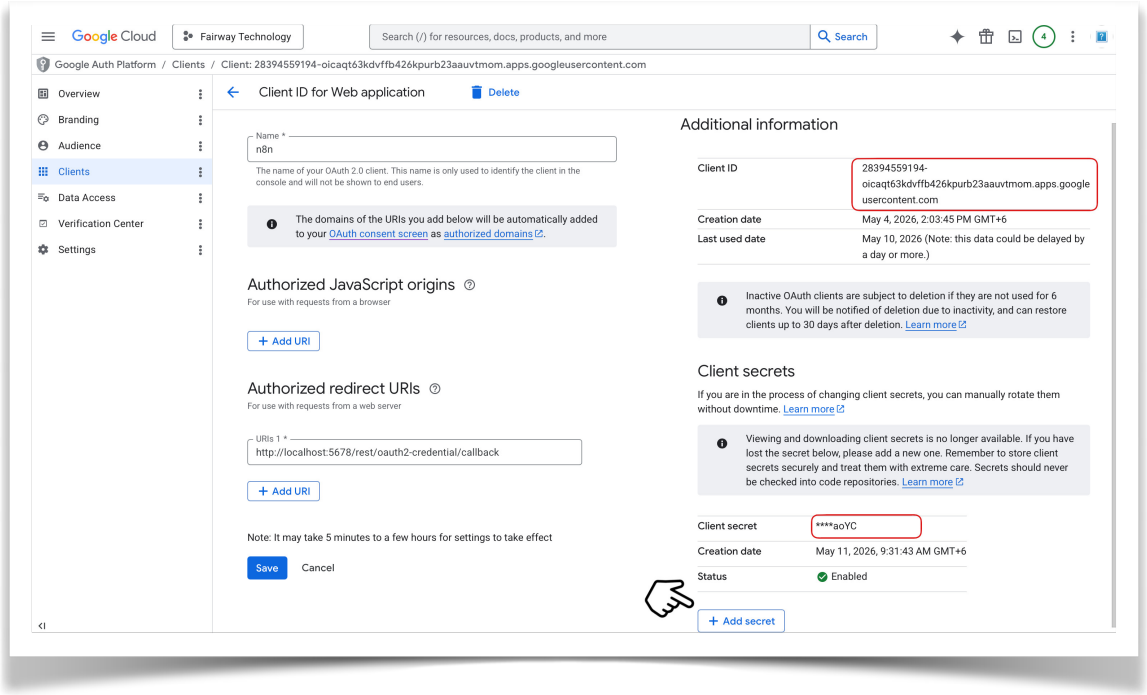




ပုံမှာပြထားသလို **Web application** ကို ရွေးပြီး အမည်ကို နှစ်သက်ရာပေးပါ။ နမူနာမှာ **n8n** လို့ပဲ ပေးထားပါတယ်။ အရေးကြီးတာက **Authorized redirect URIs** ဖြစ်ပါတယ်။ Add URI ကို နှိပ်ပြီး ဒီလင့်ကို ထည့်ပေးပါ။

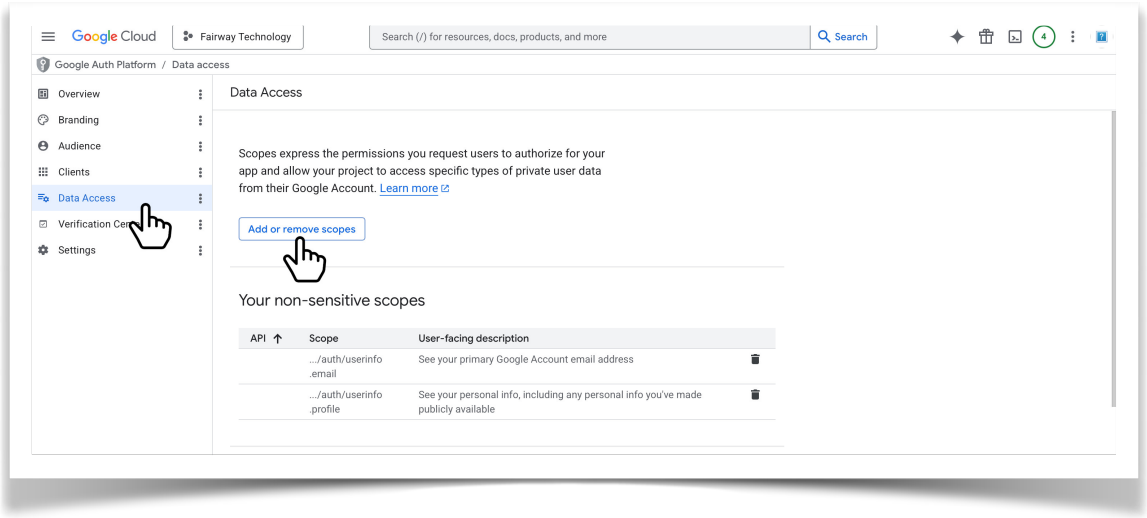
http://localhost:5678/rest/oauth2-credential/callback

ဒီတော့မှ Google API တွေက ကိုယ့်စက်ထဲမှာရှိတဲ့ n8n ကို လက်ခံအလုပ်လုပ်ပေးမှာပါ။ အချက်အလက်တွေစုံရင် **Create** နှိပ်လိုက်ပါ။



ပုံမှာ ဝိုင်းပြထားတဲ့ **Client ID** ကို ကူးယူသိမ်းဆည်းထားပါ။ **Client secret** ကိုလည်း ကူးယူသိမ်းဆည်းထားပါ။ Client secret ကို အပြည့်အစုံမပြပါဘူး။ လက်ရှိပြထားတဲ့ *** လိုဟာတွေကို ကူးယူရတာ မဟုတ်ပါဘူး။

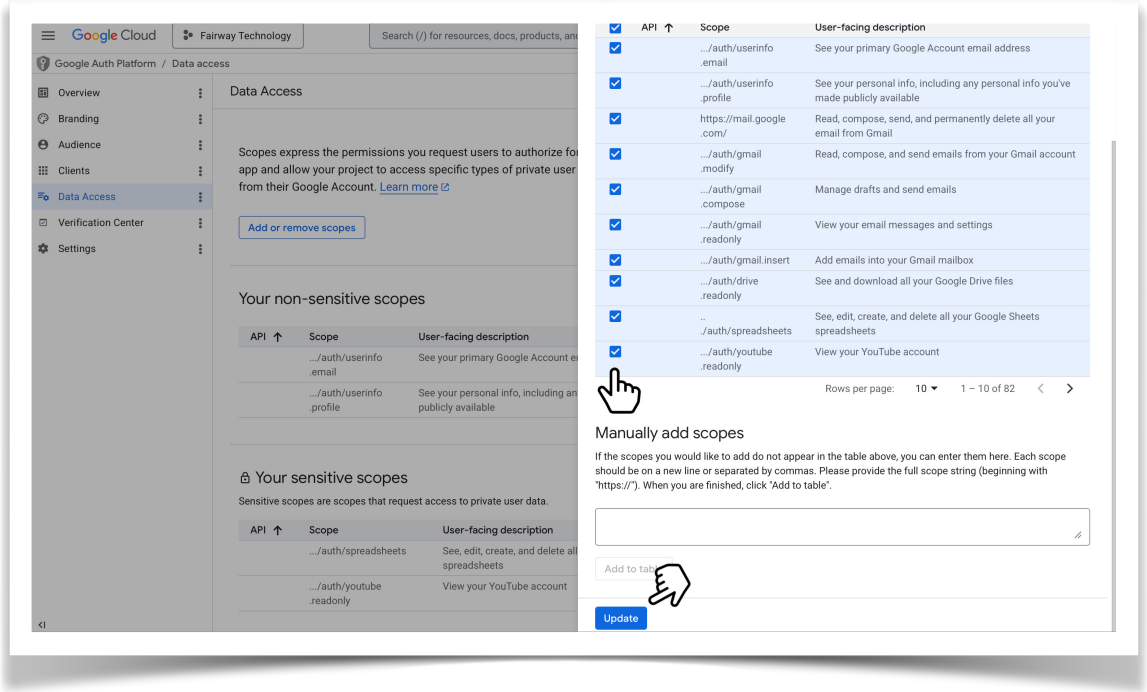
လိုအပ်ရင် **+ Add secret** ခလုတ်ကိုနှိပ်ပြီး အသစ်ပေါ်လာတဲ့ Client secret အပြည့်အစုံကို ကူးယူသိမ်းဆည်းထားပါ။ နောက်တစ်ဆင့်အနေနဲ့ **Data Access** ကို နှိပ်လိုက်ပါ။ **Add or remove scopes** ကို နှိပ်ပါ။



ပေါ်လာတဲ့ စာရင်းထဲက နဂိုရွေးထားတာတွေကို ဒီအတိုင်းထားပြီး၊ ဒီစာရင်းအတိုင်း တွေ့အောင်ရှာရွေးလိုက်ပါ။

- .../auth/gmail.modify
- .../auth/gmail.compose
- .../auth/gmail.readonly
- .../auth/gmail.insert
- .../auth/drive.readonly
- .../auth/spreadsheets
- ...../auth/youtube.readonly

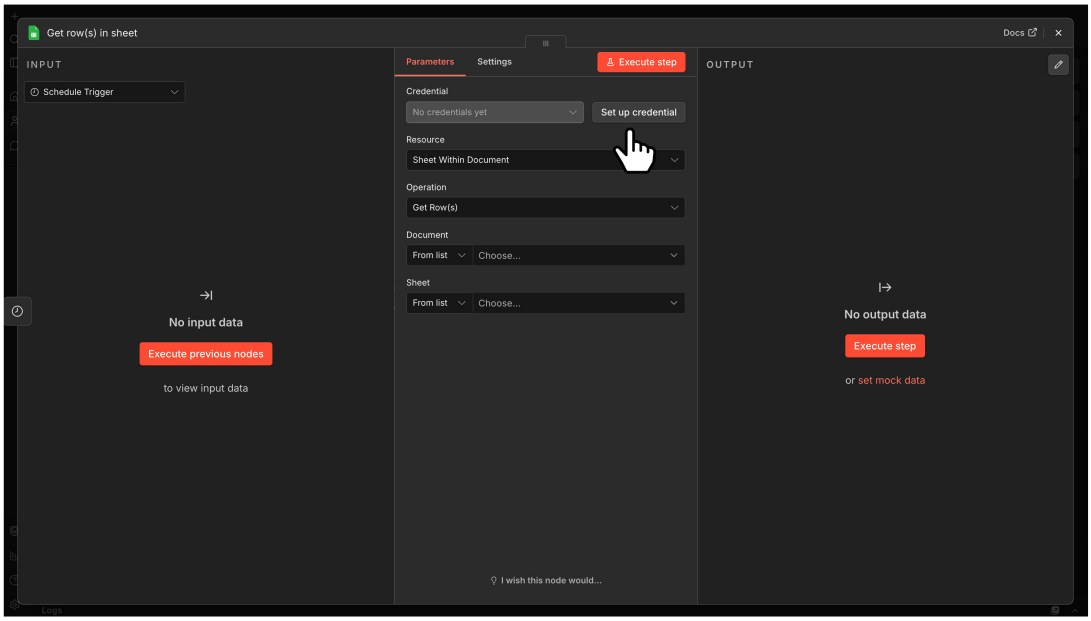
စုံအောင်ရွေးပြီးပြီဆိုရင် **Update** လုပ်လိုက်ပါ။

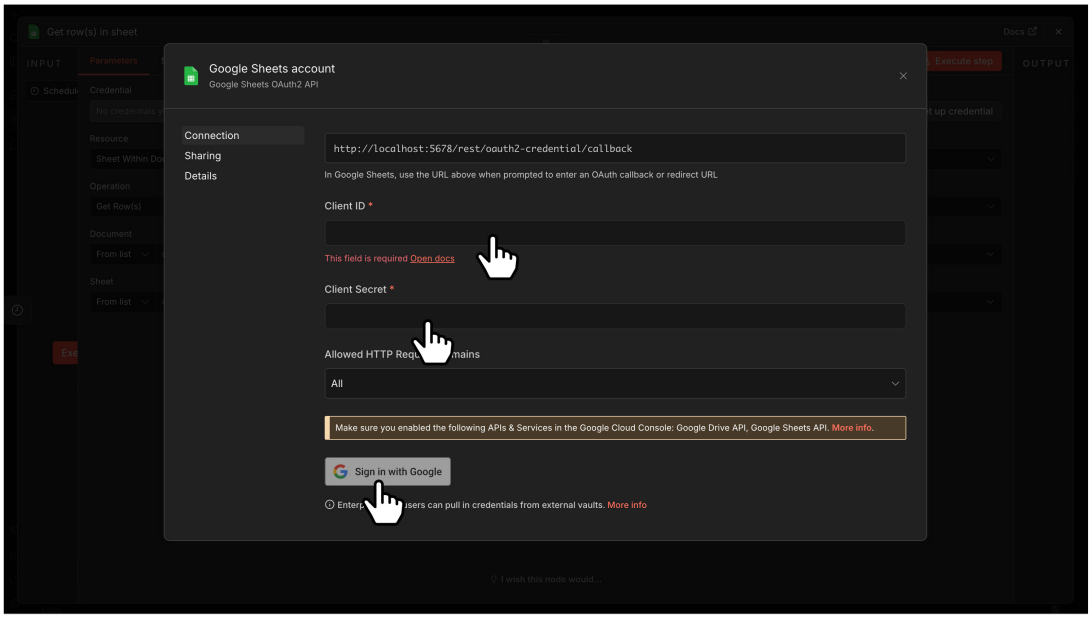


ရသွားပါပြီ။ တော်တော်အလုပ်ရှုပ်သွားပေမဲ့ ဒီအထိ စုံအောင်လုပ်ပြီးသွားရင် n8n ကနေ Google Sheets, Google Drive, Gmail နဲ့ YouTube တို့ကို လှမ်းပြီးတော့ စီမံလို့ရသွားပါပြီ။ နောက်ပိုင်းမှာ Google Doc တို့ဘာတို့ ထပ်ပြီး လိုအပ်ရင်လည်း အဲ့ဒီနည်းအတိုင်း **Add or remove scopes** မှာ ထပ်တိုးထားလို့ ရပါတယ်။

လုပ်လက်စ Workflow ကို ပြန်သွားကြပါမယ်။

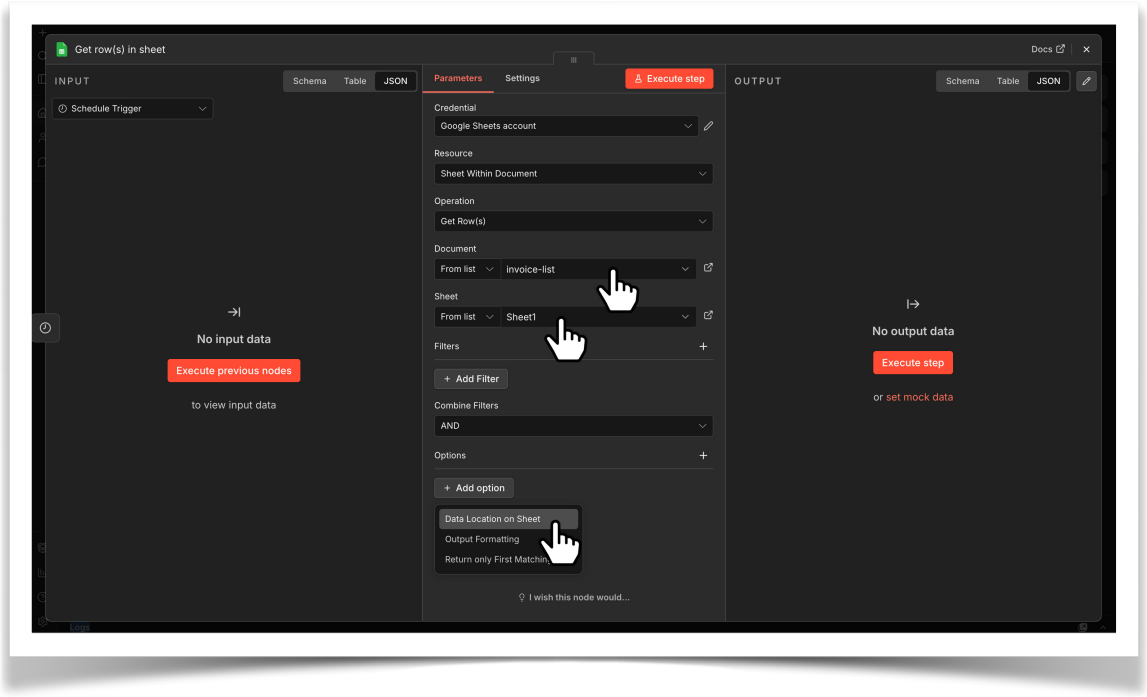
Google Sheet Node က လက်ရှိ အချက်အလက် မပြည့်စုံသေးတဲ့အတွက် Warning သင်္ကေတလေးနဲ့ ပြနေပါလိမ့်မယ်။ Double-Click လုပ်လိုက်ပြီး လိုအပ်တဲ့ အချက်အလက်တွေ ထည့်ပေးနိုင်အောင် **Set up credential** ကို နှိပ်လိုက်ပါ။


Get row(s) in sheet
Docs
INPUT
Schedule Trigger
No input data
Execute previous nodes
to view input data
Parameters
Settings
Execute step
Credential
No credentials yet
Set up credential
Resource
Sheet Within Document
Operation
Get Row(s)
Document
From list
Choose...
Sheet
From list
Choose...
I wish this node would...
OUTPUT
No output data
Execute step
or set mock data
Logs


Google Sheets account
Google Sheets OAuth2 API
Connection
Sharing
Details
http://localhost:5678/rest/oauth2-credential/callback
In Google Sheets, use the URL above when prompted to enter an OAuth callback or redirect URL
Client ID *
This field is required Open docs
Client Secret *
Allowed HTTP Req
mains
All
Make sure you enabled the following APIs & Services in the Google Cloud Console: Google Drive API, Google Sheets API. More info.
Sign in with Google
sers can pull in credentials from external vaults. More info

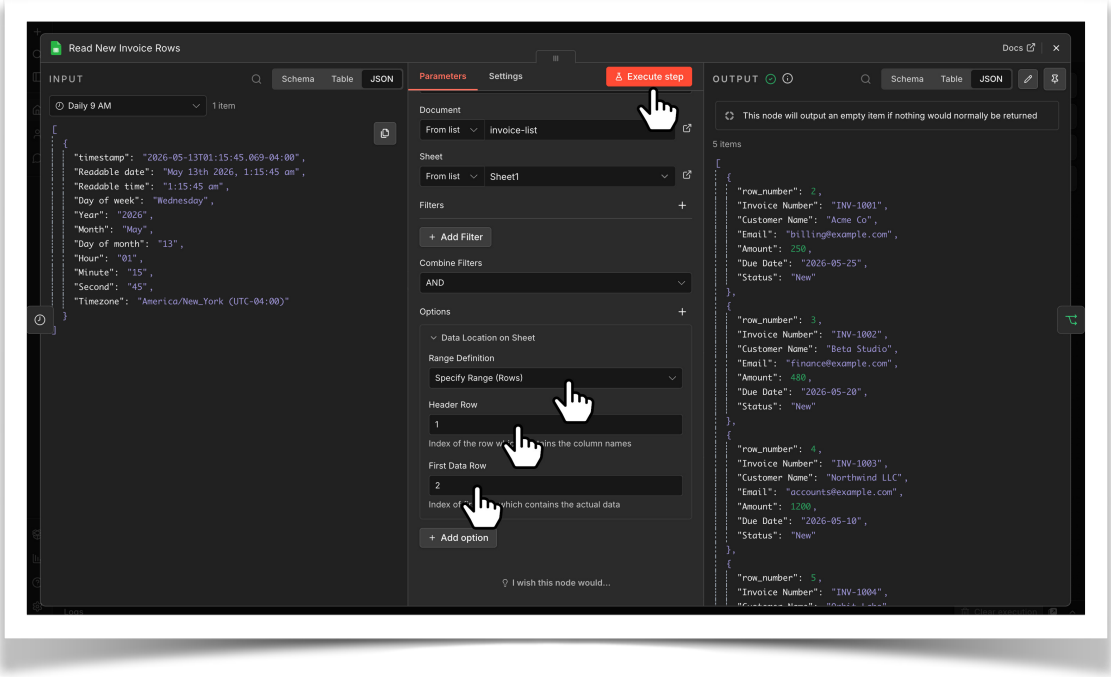
ပြီးတဲ့အခါ စောစောက Google Cloud Console ကနေ ရရှိထားတဲ့ **Client ID** နဲ့ **Client Secret** ကို ဖြည့်လိုက်ပါ။ **Sign in with Google** ခလုတ်ကို နှိပ်ကြည့်လိုက်ပါ။ အချက်အလက်တွေမှန်တယ်ဆိုရင် Google နဲ့ချိတ်မိသွားပါလိမ့်မယ်။

Credential ရပြီဆိုရင် ကိုယ့် Sheets တွေထဲက **invoice-list** နဲ့ **Sheet1** ကို ပုံမှာပြထားသလို ရွေးလိုက်ပါ။ ပြီးတဲ့အခါ **+ Add option** ကိုနှိပ်ပြီး **Data Location on Sheet** ကို ရွေးပါ။

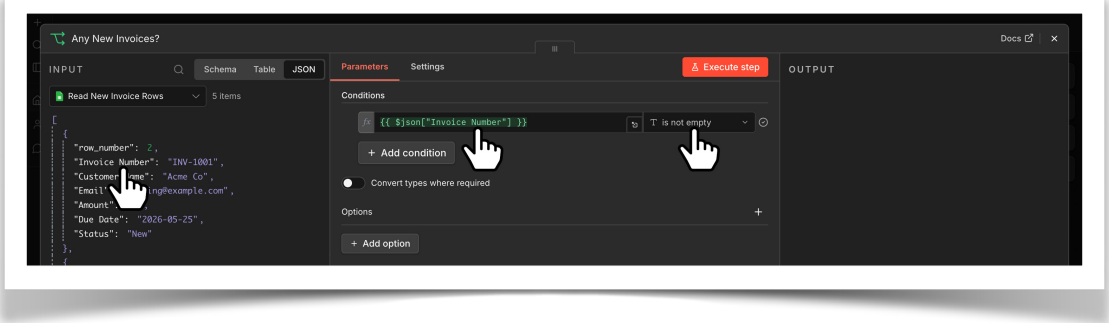


သူအလိုအလျောက်လည်း သိနိုင်ပေမဲ့ ပထမဆုံး Row က Header ဖြစ်ပြီး Data က ဒုတိယ Row ကနေ စတယ်ဆိုတာကို ပြောပြချင်လို့ပါ။

ဒါကြောင့် **Range Definition** အတွက် **Specify Range (Rows)** ကိုရွေးပါ။ **Header Row** ကို 1 လို့ပြောပြီး **First Data Row** ကို 2 လို့ ထည့်ပေးလိုက်ပါ။

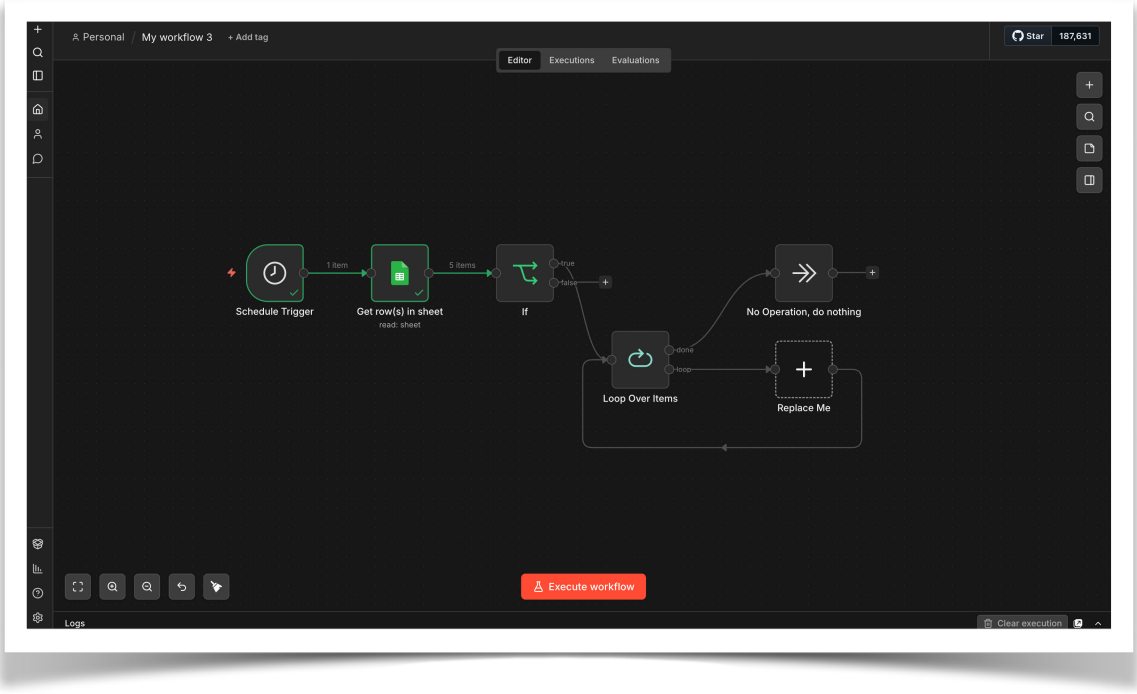


ရပါပြီ။ Google Sheets အတွက် အားလုံးအသင့်ဖြစ်သွားပါပြီ။ Run ကြည့်လိုက်ရင် Google Sheets ထဲက ဒေတာတွေကို ရရှိပါလိမ့်မယ်။ နောက်တစ်ဆင့်အနေနဲ့ If Node တစ်ခု ထပ်ထည့်ပြီး အခုလိုသတ်မှတ်ပေးလိုက်ပါ။



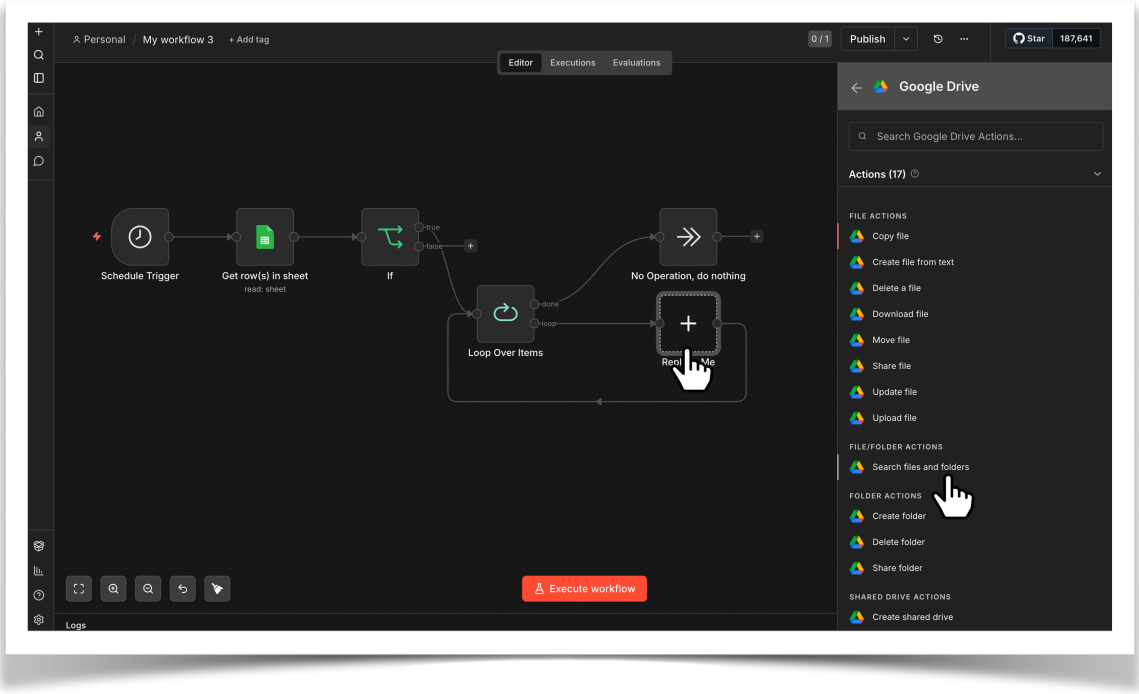
Invoice Number ကို ဆွဲထည့်လိုက်ပြီး **String → Is not empty** ကို ရွေးလိုက်ပါ။ Invoice Number ရှိမှသာ ရှေ့ဆက်အလုပ်လုပ်မယ်ဆိုတဲ့ သဘောပဲ ဖြစ်ပါတယ်။

ပြီးတဲ့အခါ **Loop Node** တစ်ခုနဲ့ **Do Nothing Node** တစ်ခု ထပ်ထည့်ပြီး အခုလိုရလဒ်ရအောင် လုပ်လိုက်ပါ။ တစ်ဆင့်ချင်း မပြောတော့ပါဘူး။ Node Parameter တွေ ပြင်စရာမလိုတဲ့အတွက် ကိုယ့်ဘာသာ လုပ်နိုင်လိမ့်မယ်လို့ ယူဆပါတယ်။



If Node ရဲ့ **true** မှာ **Loop Node** ကို ချိတ်ပေးထားပြီး **Loop Node** ရဲ့ **done** မှာ **Do Nothing** ကို ချိတ်ပေးလိုက်တာပါ။ နောက်က **Replace Me** က Loop Node ကို ထည့်လိုက်ရင် သူ့ဘာသာ အလိုအလျောက် ပါဝင်သွားပါလိမ့်မယ်။

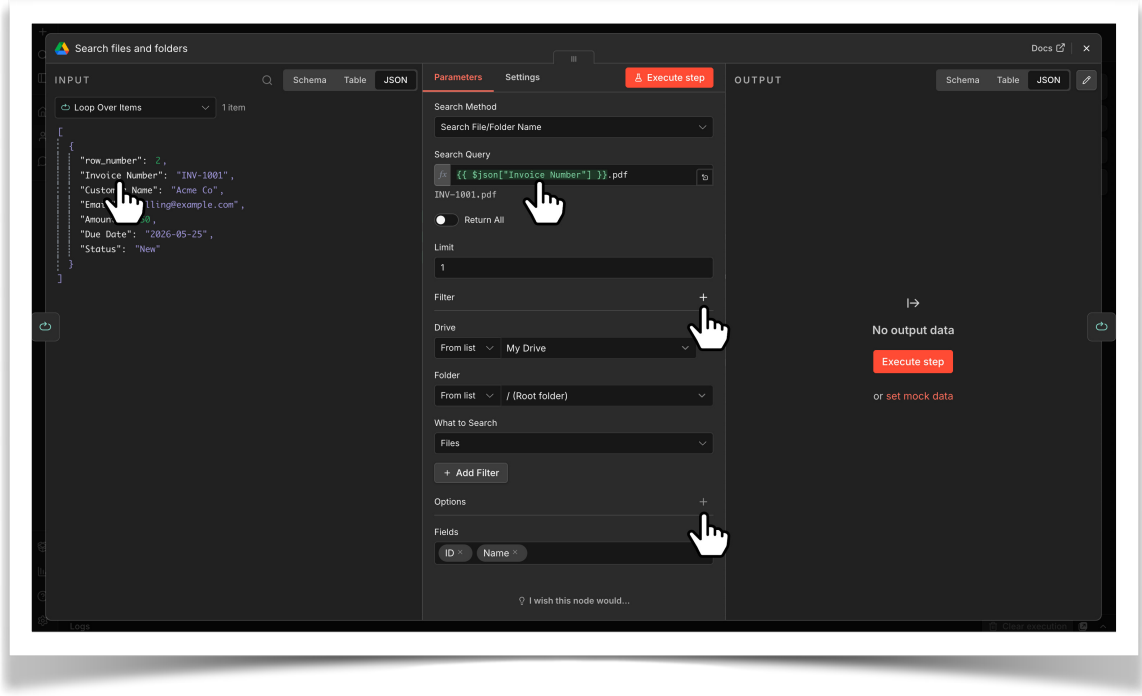
Loop Node ကို ပုံစံတူအလုပ်တွေ ထပ်ခါထပ်ခါ လုပ်ချင်တဲ့အခါ အသုံးပြုပါတယ်။ Invoice List ရပြီဖြစ်လို့ Email တွေကို တစ်ဦးချင်း ထပ်ခါထပ်ခါ ပေးပို့တော့မှာ မို့လို့ပါ။ ပေးပို့တဲ့အခါ Invoice ဖိုင်ကို Attachment အနေနဲ့ တွဲပြီး ပို့ချင်တဲ့အတွက် ဒီတစ်ခါ **Drive API** ကို အသုံးပြုကြပါမယ်။ **Replace Me** ကို Double-Click လုပ်လိုက်ပါ။



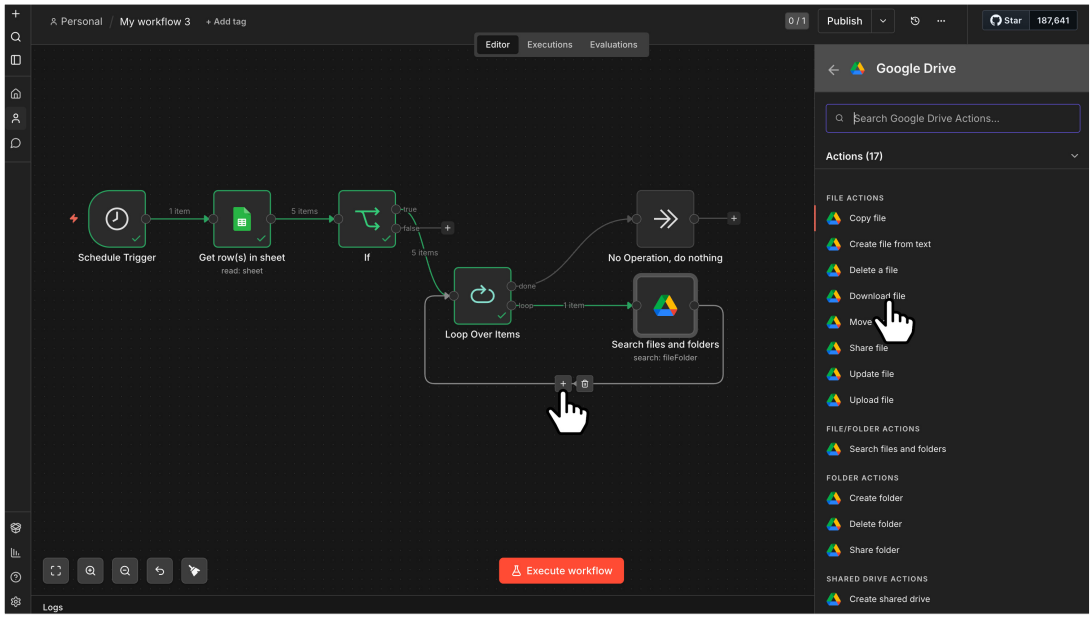
Search files and folders Node ကို ထည့်လိုက်တာပါ။ စောစောက Google Sheets အတွက် Credential ထည့်ရသလိုပဲ Drive အတွက်လည်း ထည့်ရပါမယ်။ ပုံစံတူပဲမို့လို့ ထပ်မပြတော့ပါဘူး။ **Client ID** နဲ့ **Client Secret** ကိုပဲထည့်ရတာပါ။

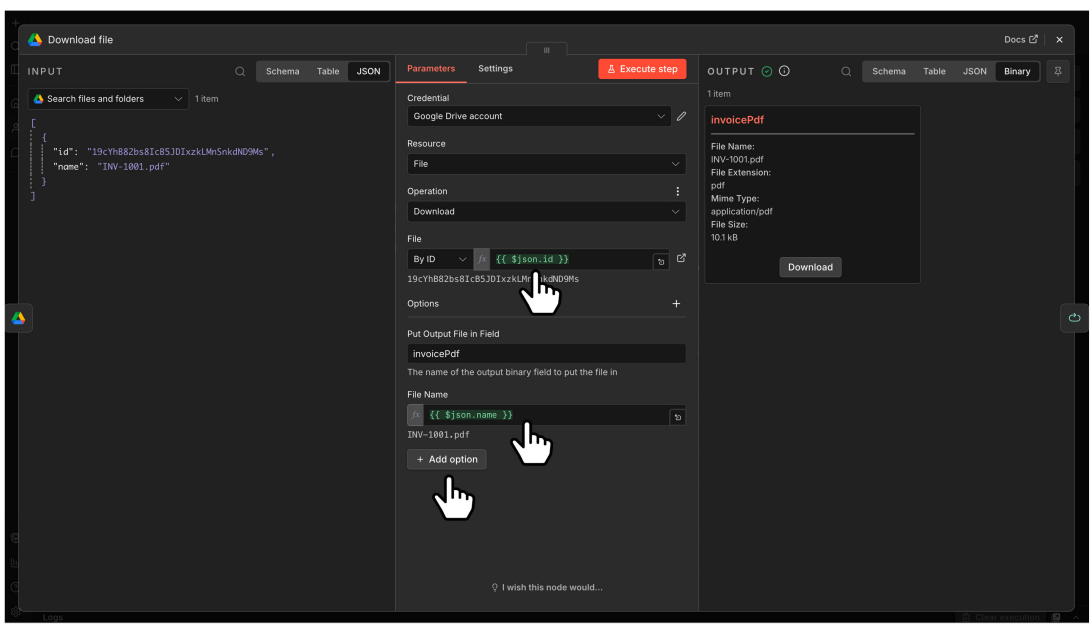
Credential ထည့်ပြီးရင် နောက်တစ်ဆင့်ကပုံမှာ ပြထားတဲ့အတိုင်း Parameter တွေ စုံအောင် ထည့်ပေးပါ။ Invoice Number ကို File Name အဖြစ်သုံးထားတဲ့ Invoice PDF

ဖိုင်ကို ရှာမှာ ဖြစ်ပါတယ်။ Filter တွေ Option တွေ ပုံကိုကြည့်ပြီးတော့ စုံအောင် ထည့်ပေးလိုက်ပါ။ လက်တွေ့စမ်းချင်ရင် Invoice Number နဲ့ PDF ဖိုင်တွေကတော့ Drive ထဲမှာ အမှန်တကယ် ရှိနေဖို့လိုပါတယ်။ အဲဒီအပိုင်းကိုတော့ ကိုယ်တိုင်ပဲ လုပ်ထားရမှာပါ။

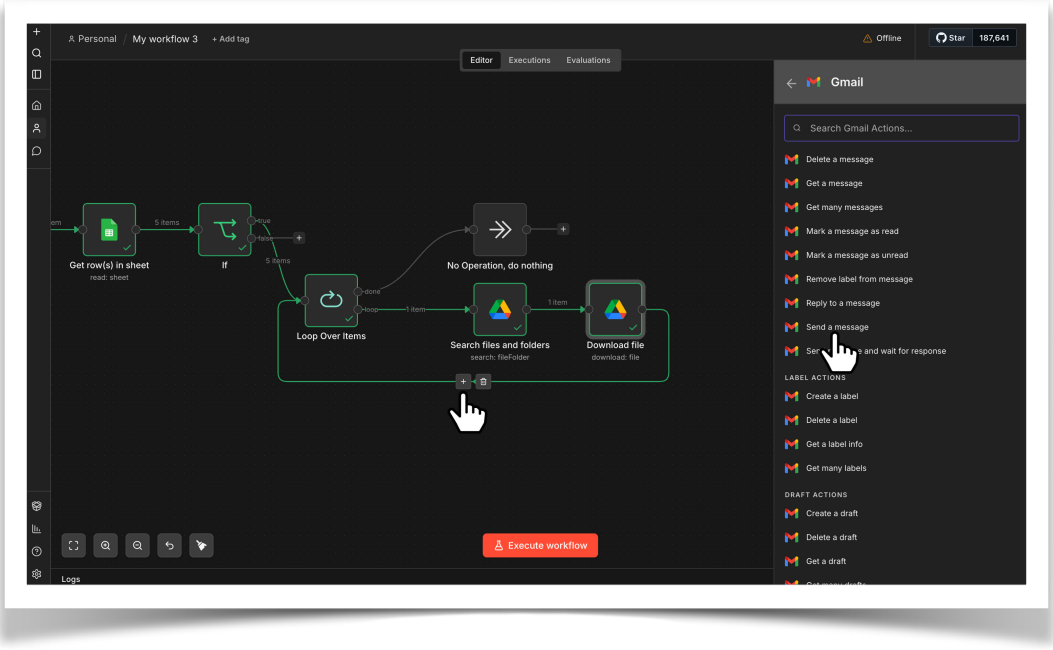


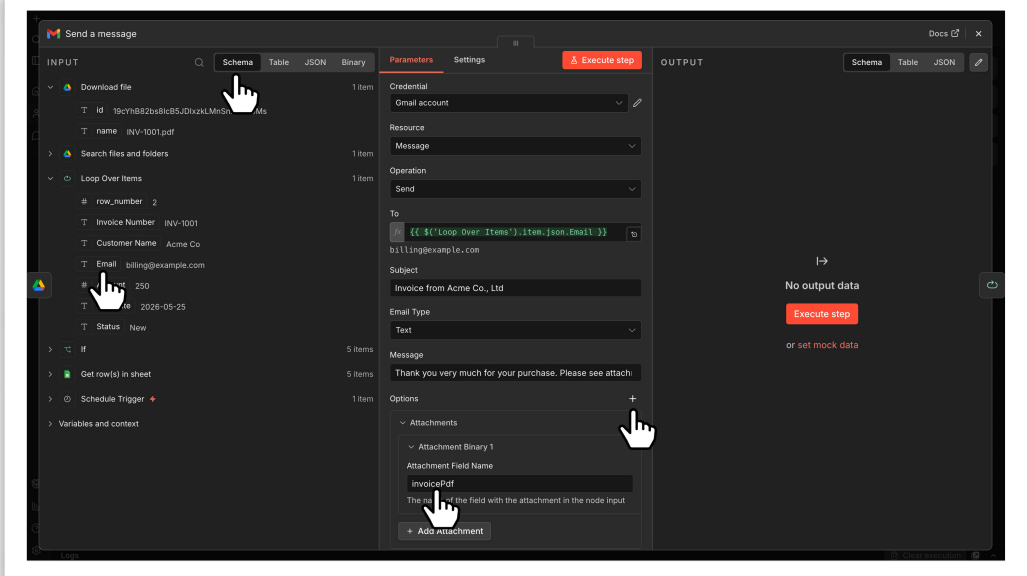
ပြီးတဲ့အခါ နောက်ထပ် **Drive Node** တစ်ခုထပ်ထည့်ပါ။


Personal / My workflow 3 + Add tag
Editor Executions Evaluations
Publish
Star 187,641
Schedule Trigger
Get row(s) in sheet
read: sheet
If
No Operation, do nothing
Loop Over Items
Search files and folders
search: fileFolder
Execute workflow
Logs
Google Drive
Search Google Drive Actions...
Actions (17)
FILE ACTIONS
Copy file
Create file from text
Delete a file
Download file
Move file
Share file
Update file
Upload file
FILE/FOLDER ACTIONS
Search files and folders
FOLDER ACTIONS
Create folder
Delete folder
Share folder
SHARED DRIVE ACTIONS
Create shared drive


Download file
Docs
INPUT
Schema Table JSON
Search files and folders
1 item
Parameters Settings
Execute step
Credential
Google Drive account
Resource
File
Operation
Download
File
By ID
{{ $json.id }}
Options
Put Output File in Field
invoicePdf
The name of the output binary field to put the file in
File Name
{{ $json.name }}
INV-1001.pdf
+ Add option
I wish this node would...
OUTPUT
Schema Table JSON Binary
1 item
invoicePdf
File Name:
INV-1001.pdf
File Extension:
pdf
Mime Type:
application/pdf
File Size:
10.1 kB
Download

Google Drive Node နှစ်ခုမှာ ပထမတစ်ခုက ဖိုင်ကိုရှာပေးပြီး၊ ဒုတိယတစ်ခုကတော့ ဖိုင်ကို Download ယူလိုက်တာပါ။ နောက်တစ်ဆင့်မှာ **Gmail Node** တစ်ခုထည့်လိုက်ပါ။

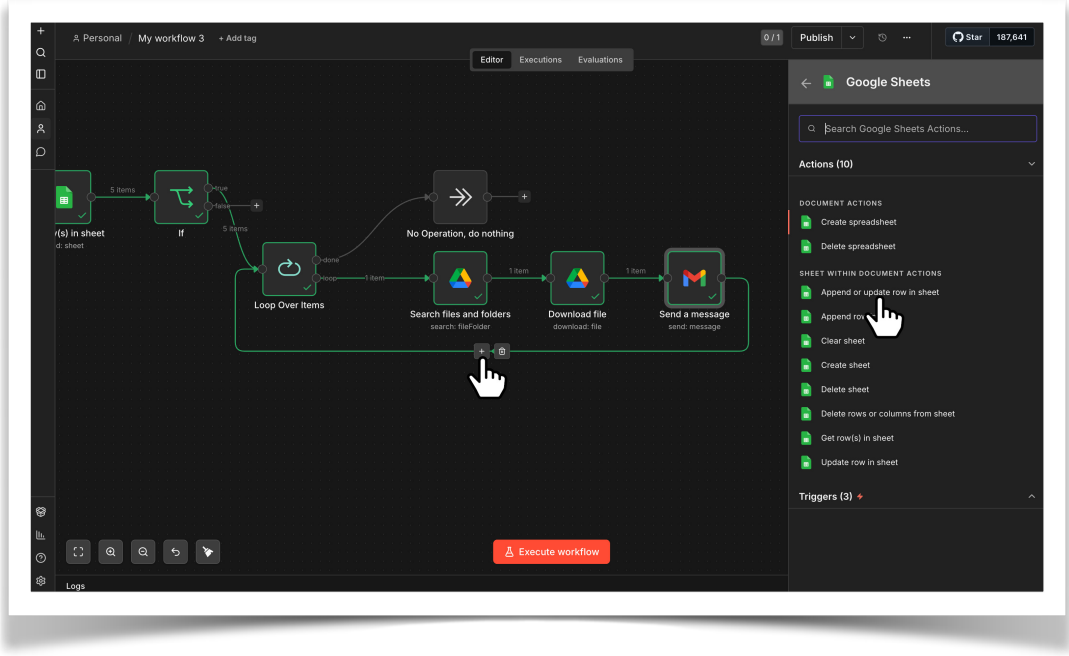


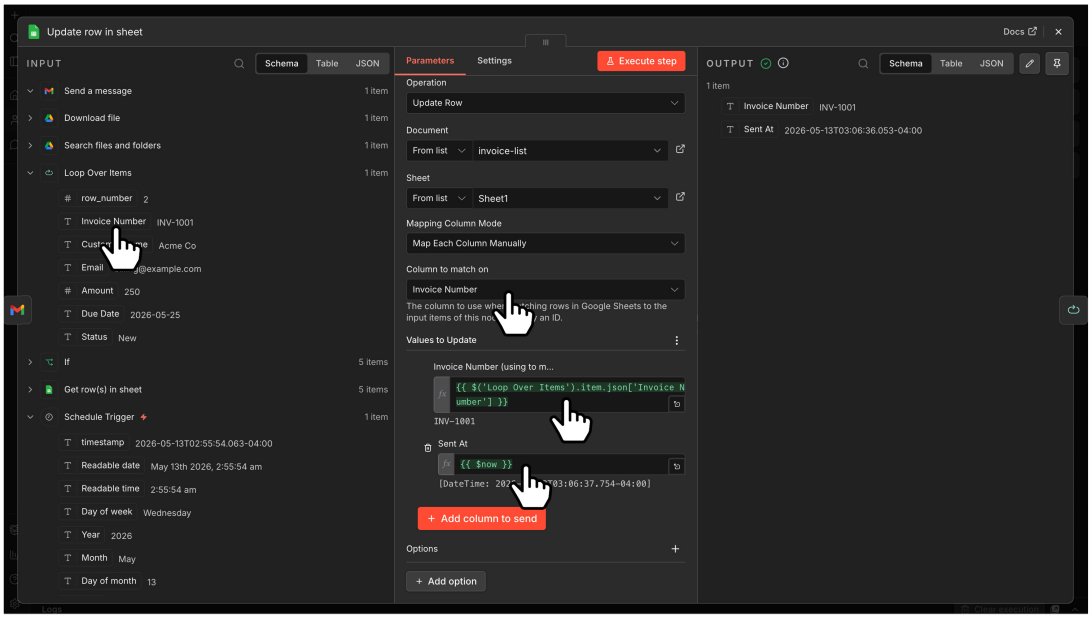


Gmail အတွက်လည်း Credential ထည့်ပေးဖို့လိုပါတယ်။ ထည့်ပုံထည့်နည်း Sheets တို့ Drive တို့နဲ့ အတူတူပါပဲ။ ဒီအဆင့်မှာ Input ကို JSON မရွေးဘဲ **Schema** ရွေးထားတာ သတိပြုပါ။ ဒီတော့မှ ဟိုးရှေ့အဆင့်တွေက တခြားအချက်အလက်တွေကိုလည်း ကြည့်လို့ရမှာ မို့လို့ပါ။

Loop ဆီကနေရတဲ့ Email ကို To မှာ ဆွဲထည့်ပေးလိုက်ပါ။ ဒါကြောင့် သက်ဆိုင်ရာ Customer Email တစ်ခုချင်းစီကို ပို့ပေးသွားမှာ ဖြစ်ပါတယ်။ **Subject** နဲ့ **Message** မှာ စမ်းသပ်ရုံပဲမို့လို့ မိမိနှစ်သက်ရာ ပေးပို့လိုတဲ့ စာတွေကို ထည့်ပေးလိုက်ပါ။

အောက်ဆုံးမှာသာ ပုံမှာ ပြထားသလို **Attachment** အနေနဲ့ **invoicePdf** ကို ချိတ်ပေးလိုက်ရမှာ ဖြစ်ပါတယ်။ ပြီးသလောက်ဖြစ်သွားပါပြီ။ နောက်ဆုံးတစ်ဆင့်အနေနဲ့ Email ပေးပို့ပြီးကြောင်း Sheets မှာ မှတ်တမ်းသွင်းဖို့ ထပ်ထည့်လိုက်ပါ။





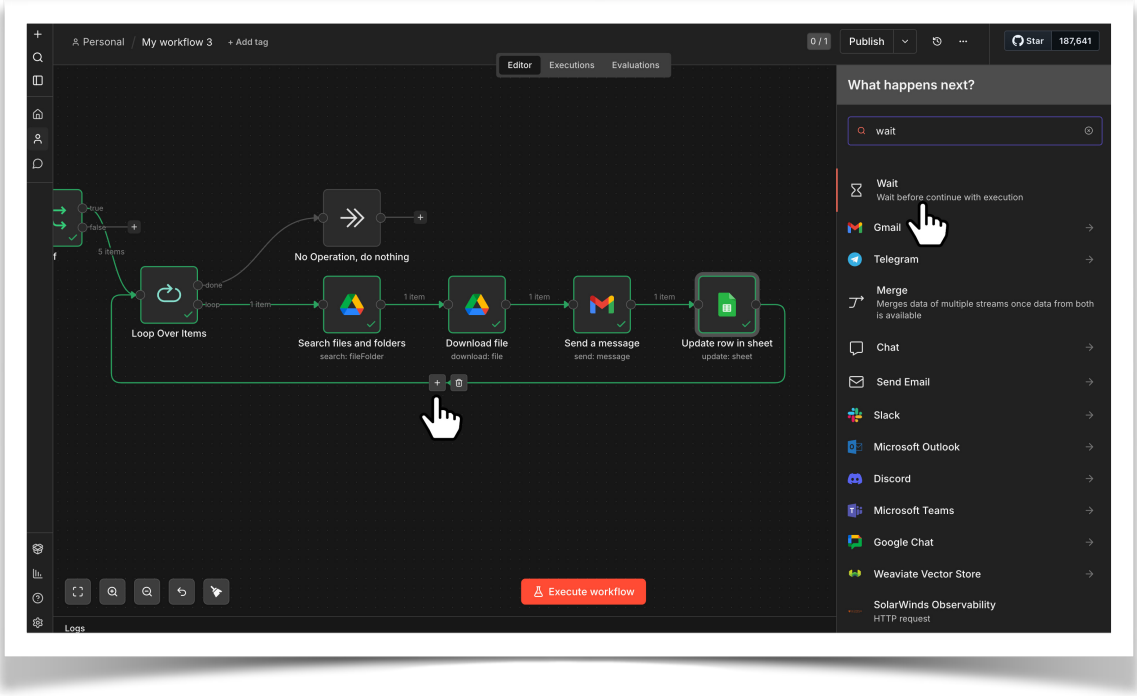
ပုံမှာပြထားတဲ့အတိုင်း Parameter တွေ မှန်အောင်ထည့်လိုက်ပါ။ Credential ကတော့ ထည့်ပြီးသားမို့လို့ ထပ်ထည့်စရာ မလိုတော့ပါဘူး။ **Column to match on** မှာ Invoice Number ကို ရွေးလိုက်ပါ။

Invoice Number (using to match) အတွက် ဘယ်ဘက်ခြမ်းက Invoice Number ကို ဆွဲထည့်ပေးလိုက်ပါ။ Invoice Number နဲ့ တိုက်စစ်ပြီး Update လုပ်မယ်လို့ ပြောလိုက်တာပါ။ ပြီးတဲ့အခါ Update မလုပ်ချင်တဲ့ တခြား Column တွေကို ဖယ်ထုတ်လိုက်ပြီး **Sent At** မှာ `{{ $now }}` လို့ ရိုက်ထည့်ပေးလိုက်ပါ။ **Sent At** နေရာမှာ လက်ရှိ ရက်စွဲ/အချိန်ကို ထည့်ပြီး Update လုပ်မယ်လို့ ပြောလိုက်တာပါ။

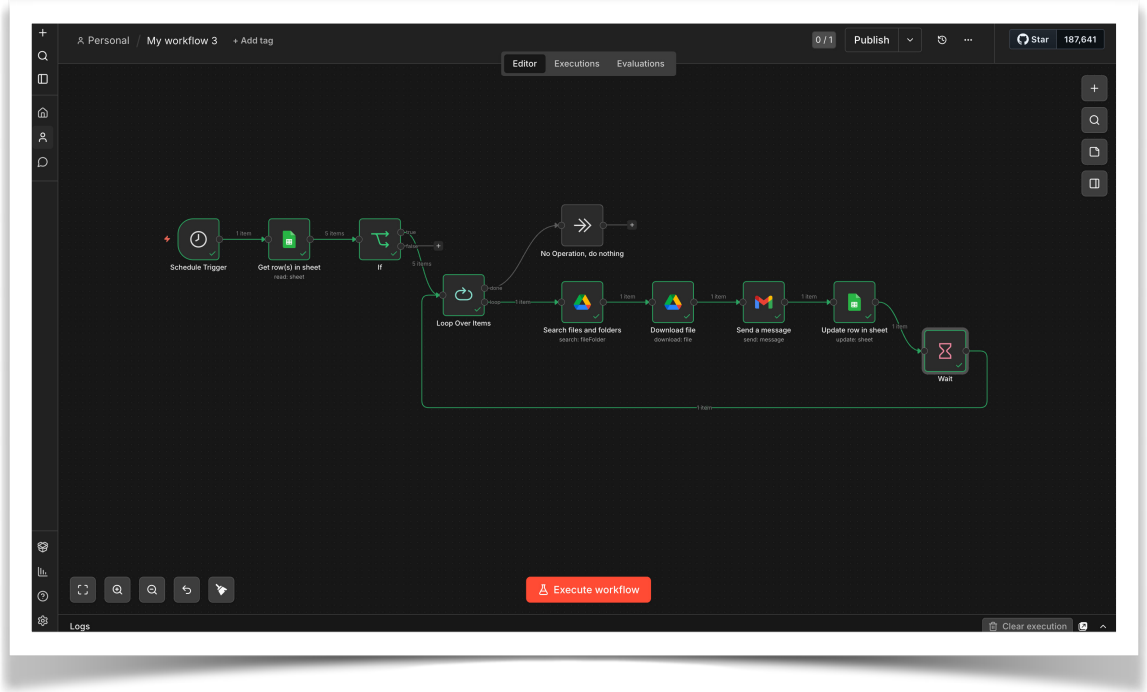
အများပြင်ဆင်ချက် - `{{ $now }}` အစား `{{ $now.toISO() }}` လို့ထည့်ပေးပါ။

ဒါကြောင့် Invoice ကို Email ပို့ပြီးတာနဲ့ Sheets မှာ Sent At အတွက် လက်ရှိရက်စွဲ အချိန်နဲ့ အလိုအလျောက် Update လုပ်ပေးသွားမှာပဲ ဖြစ်ပါတယ်။

Workflow က ပြည့်စုံသွားပါပြီ။ ဒါပေမဲ့ Email တွေကို ဆက်တိုက် မပေးပို့စေလိုဘဲ (၅) စက္ကန့်လောက်ခွာပြီးမှာ ပေးပို့စေလိုတဲ့အတွက် နောက်ထပ် Node လေးတစ်ခုလောက် ထပ်ထည့်လိုက်ပါဦး။

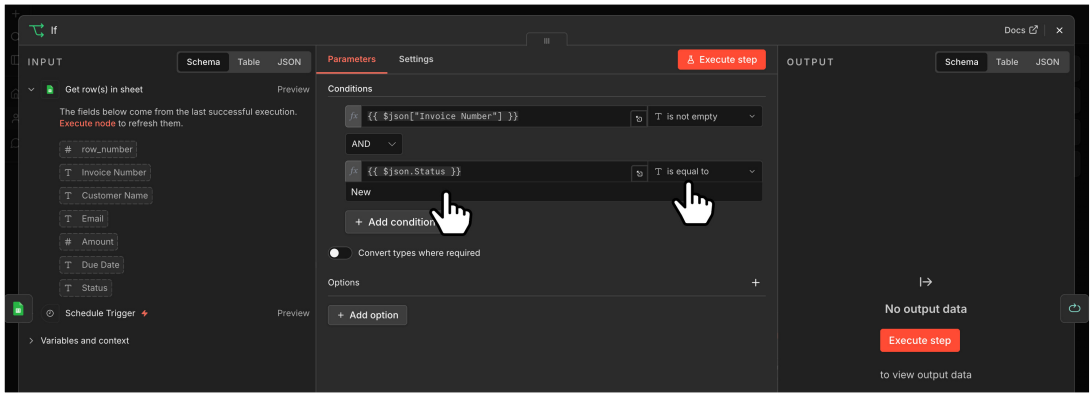


ဒီအထိ ရသွားပြီဆိုရင် နေ့စဉ်မနက် (၉) နာရီတိုင်း Invoice တွေ အလိုအလျောက် ပို့ပေးတဲ့ Workflow တစ်ခုကို အခုလို အောင်မြင်စွာ ရရှိသွားပြီပဲ ဖြစ်ပါတယ်။

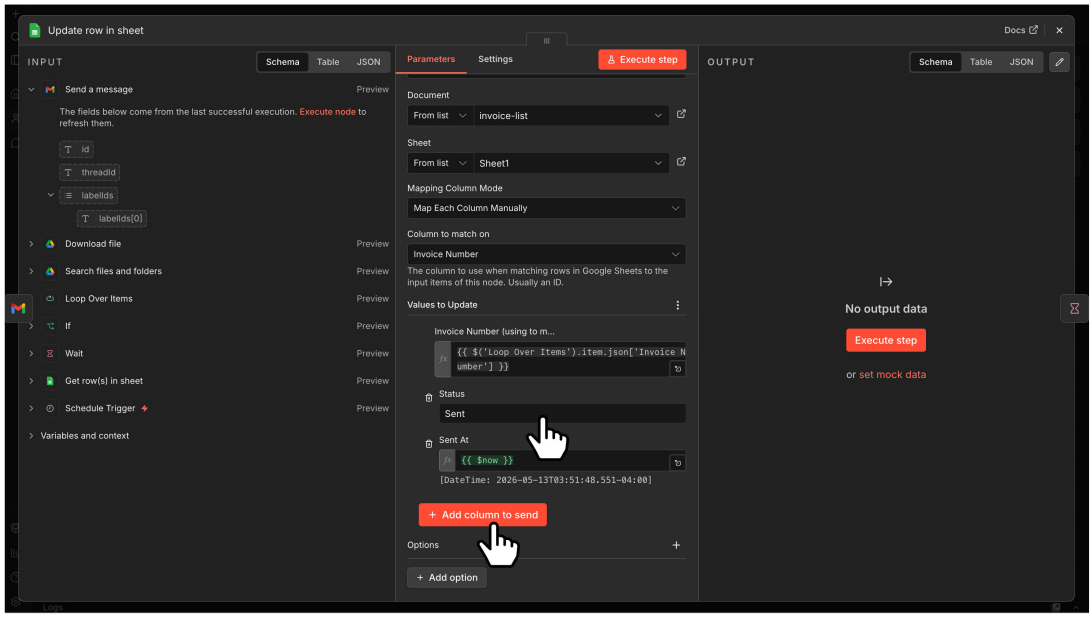


အမှားပြင်ဆင်ချက်

ဒီ Workflow ကို လက်တွေ့စမ်းသပ်ကြည့်တဲ့အခါ လိုအပ်ချက်လေးတစ်ခုကို သွားတွေ့ပါတယ်။ **If Node** မှာ Invoice Number ရှိမရှိ စစ်ပြီး အလုပ်လုပ်ထားပါတယ်။ ဒါကြောင့် Invoice Number ရှိတာနဲ့ ပို့ပြီးသား Customer တွေကိုလည်း နေ့စဉ် Invoice Email တွေ ဆက်ပို့နေမှာပါ။ အဲ့ဒါလေး ပြင်ပေးဖို့ လိုပါတယ်။ **If Node** ကို အခုလိုပြင်ပေးလိုက်ပါ။



တစ်လက်စတည်း Email ပို့ပြီးကြောင်း Update လုပ်တဲ့ Sheets ကိုလည်း အခုလို ပြင်ပေးဖို့လိုပါတယ်။ မူလက **Sent At** ကို {{ $now }} လို့ Update လုပ်ထားရာကနေ **Status** မှာလည်း **Sent** လို့ Update လုပ်ပေးလိုက်တာပါ။

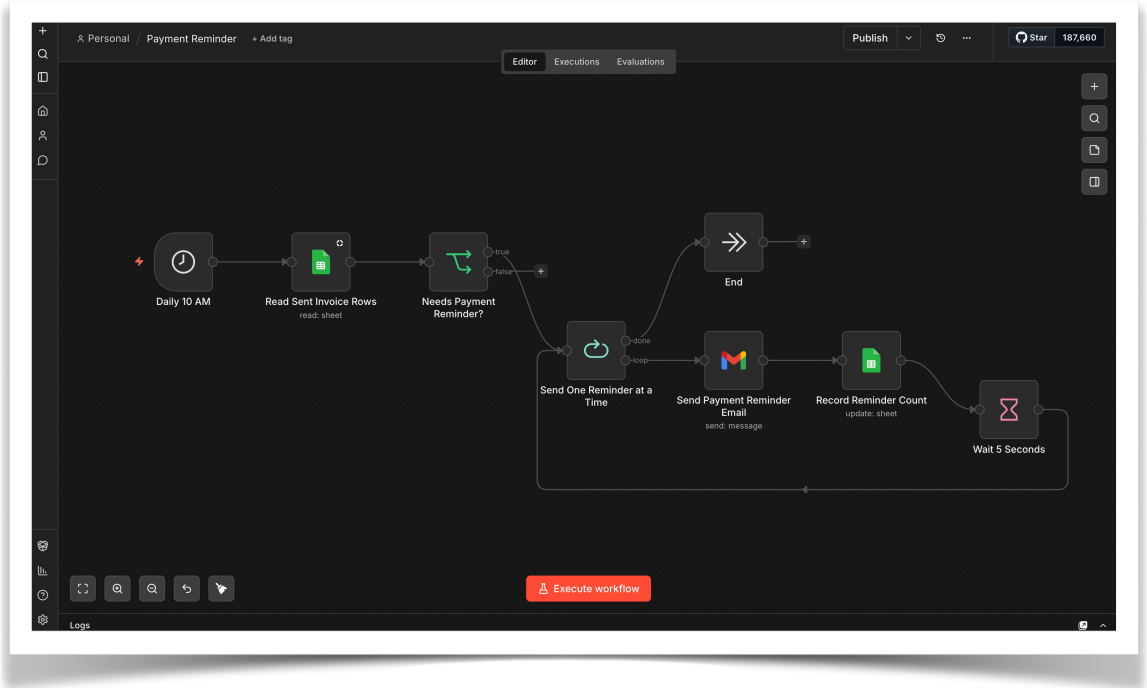


အမှားပြင်ဆင်ချက် - `{{ $now }}` အစား `{{ $now.toISO() }}` လို့ထည့်ပေးပါ။

ဒါကြောင့် Workflow က ပိုမှန်သွားပါပြီ။ နေ့စဉ် Invoice တွေပို့တဲ့အခါ စာရင်းထဲ ရှိသမျှ အကုန်မပို့တော့ဘဲ၊ New Invoice တွေကိုပဲ ပို့တော့မှာ ဖြစ်ပါတယ်။

Payment Reminder Workflow

နောက်တစ်ဆင့်အနေနဲ့ Invoice ပို့ထားပြီး ဖြစ်ပေမဲ့ Payment မရသေးရင် Reminder ပို့ဖို့လိုပါတယ်။ ဒီ Workflow ကိုတော့ တစ်ဆင့်ချင်း လုပ်မပြတော့ပါဘူး။ နည်းလမ်း ဆင်တူတဲ့အတွက် Workflow အသစ်တစ်ခုနဲ့ ကိုယ်တိုင်လုပ်ကြည့်သင့်ပါတယ်။ ဒီလိုပါ...

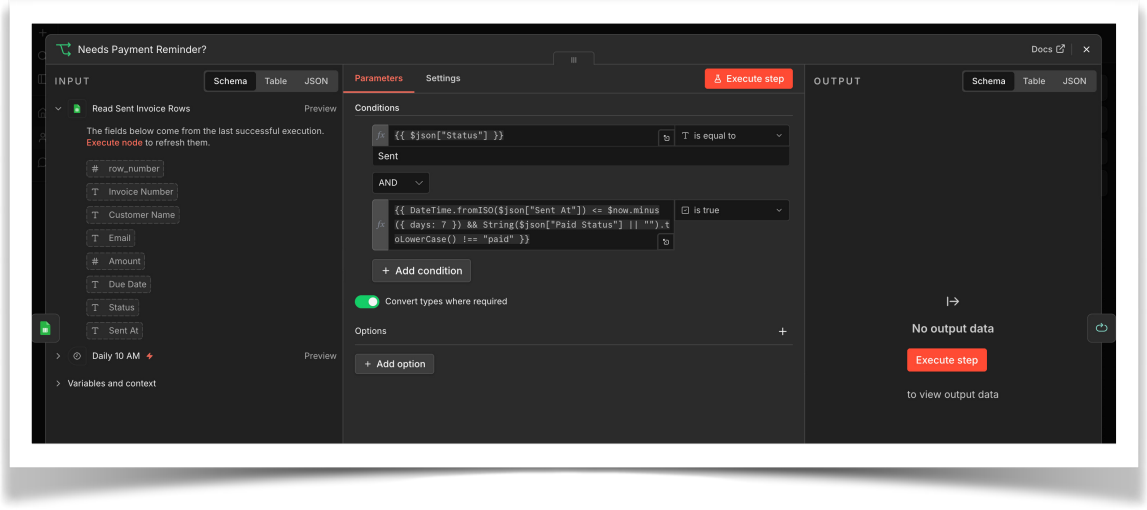


ထူးခြားချက်လေးတွေကိုတော့ ပြောပြပါမယ်။

၁။ ပထမဆုံး **Schedule Trigger** နဲ့စပါ။ နမူနာမှာ နေ့စဉ် မနက် (၁၀) နာရီရောက်တိုင်း လုပ်မယ်လို့ သတ်မှတ်ထားပါတယ်။

၂။ Google Sheets ကနေ Invoice စာရင်းကို ရယူရပါမယ်။ ဒီအထိ ပြီးခဲ့တဲ့အဆင့်မှာ လုပ်ခဲ့တဲ့ Workflow နဲ့ အတူတူပါပဲ။

၃။ **If Node** နဲ့ Invoice ပေးပို့ထားတာ (၇) ရက်ကျော်ပြီးလား စစ်ရပါမယ်။ ဒီလိုပါ...



အရင်ဆုံး **Status** က **Sent** ဖြစ်ရမယ်လို့ စစ်ထားပါတယ်။ တခြား Invoice တွေဆိုရင် အလုပ်မလုပ်ပါဘူး။ ပြီးတဲ့အခါ **Sent At** ရက်စွဲက (၇) ကျော်နေပြီးလား စစ်ပါတယ်။ ကုဒ်တွေ ရေးစရာမလိုအောင် အတတ်နိုင်ဆုံး ကြိုးစားကြည့်ပေမဲ့ ဒီတစ်ခုမှာတော့ မဖြစ်မနေ ရေးဖို့ လိုသွားပါတယ်။ ဒီကုဒ်လေး ထည့်ပေးလိုက်ပါ။

```
{{ DateTime.fromISO($json["Sent At"]) <= $now.minus({ days: 7 })
&& String($json["Paid Status"] || "").toLowerCase() !== "paid" }}
```

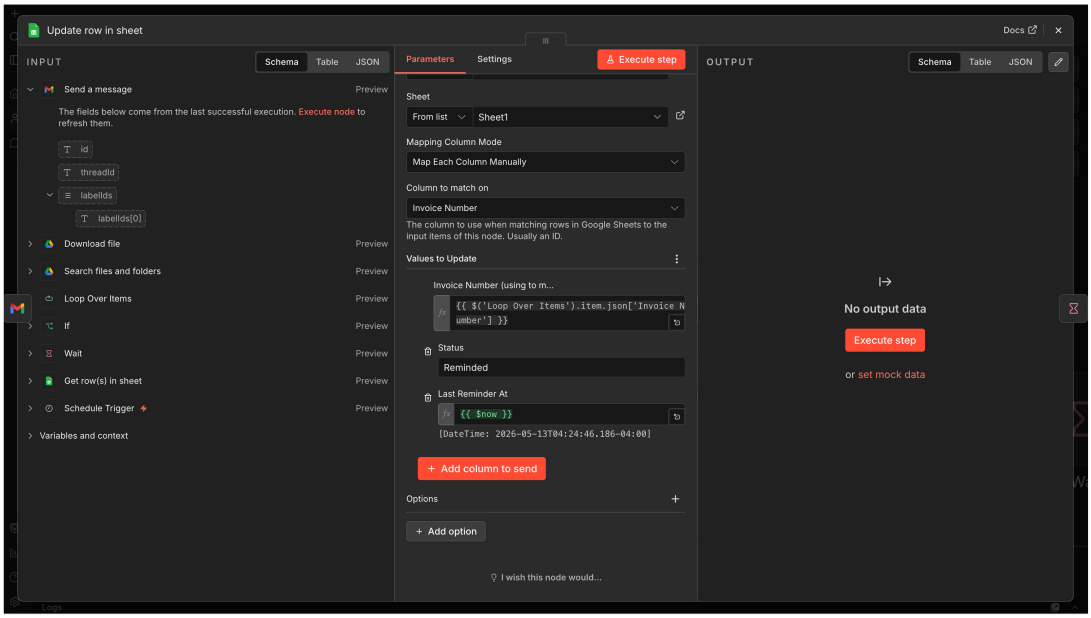
ဒီကုဒ်ရဲ့နောက်မှာ အဖြေက **Boolean → is true** ဖြစ်ရမယ်လို့ ရွေးပေးထားပါတယ်။ ဒါကြောင့် Sent လည်း ဖြစ်မယ် (၇) ရက်လည်း ကျော်ပြီဆိုမှ အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။

မှတ်ချက် - ကုဒ်တွေမဖြစ်မနေရေးဖို့ လိုသွားပေမဲ့ ကိုယ်တိုင်မရေးတတ်ရင် AI ကို ရေးခိုင်းလို့ရပါတယ်။ n8n အတွက်လို့ပြောပြီး ကိုယ်လိုချင်တာ ပြောပြလိုက်ရင် ရေးပေးပါလိမ့်မယ်။ ကူးယူထည့်သွင်းလိုက်လို့ ရနိုင်ပါတယ်။

၄။ ပြီးတဲ့အခါ စောစောက Workflow လိုပဲ **Loop Node** တစ်ခုထည့်ရပါမယ်။

၅။ **Gmail** နဲ့ စောစောကလိုပဲ Email ပို့ရပါမယ်။ ဒီတစ်ခါ Attachment တွေ Invoice ဖိုင်တွေ ထပ်မထည့်တော့ပါဘူး။ ဒါကြောင့် Google Drive ကို မသုံးတော့ပါဘူး။ ပြီးခဲ့တဲ့ Workflow ကလိုပဲ Drive နဲ့ချိတ်ပြီး Invoice ဖိုင်ကို ထပ်ထည့်ချင်ရင် လည်းရနိုင်ပါတယ်။

၆။ ပြီးတဲ့အခါ **Sheets** မှာ Update လုပ်ရပါမယ်။ ဒီလိုပါ...



Status ကို **Reminded** လို့ ပြောင်းလိုက်ပြီး **Last Reminder At** ကို {{ $now }} လို့ ထည့်ပေးလိုက်တာပါ။

အများပြင်ဆင်ချက် - `{{ $now }}` အစား `{{ $now.toISO() }}` လို့ထည့်ပေးပါ။

၇။ နောက်ဆုံးမှာ (၅) စက္ကန့်စောင့်ပြီးမှ ထပ်အလုပ်လုပ်ဖို့ Wait Node ကို ထည့်ရပါမယ်။

မူလစဉ်းစားထားတဲ့ထဲမှာ **Reminder Count** လို့ ကိစ္စမျိုးတွေလည်း ပါပါတယ်။ Reminder အီးမေးလ်ကို တစ်ကြိမ်တည်းမဟုတ်ဘဲ အကြိမ်ကြိမ် ပို့ပြီး ဘယ်နှစ်ခါပို့ပြီး ပြီလဲဆိုတာပါ ထည့်မှတ်ဖို့ပါ။

နောက်ထပ် **If Node** တစ်ခုထပ်တိုးပြီး လုပ်ရင်ရနိုင်ပေမဲ့ အရမ်းရှုပ်သွားမှာစိုးလို့ အဲ့ဒီအဆင့်ကိုတော့ ချန်ခဲ့လိုက်ပါ။

ဒီအထိရပြီဆိုရင် Invoice Workflow နဲ့အတူ ပူးတွဲအသုံးပြုဖို့အတွက် Payment Reminder Workflow ကိုပါရရှိသွားတာပါ။ ကိုယ့်ဘာသာ ကြိုးစားပြီး လုပ်ကြည့်ပါ။ လိုအပ်ရင်တော့ ကြိုလုပ်ပေးထားတဲ့ နမူနာ Workflow ဖိုင်တွေကို အောက်ပါလင့်မှာ ရယူပြီး လိုအပ်တဲ့ Credential ထည့်စမ်းကြည့်လို့လည်း ရပါတယ်။

https://github.com/eimg/n8n-workflows/archive/refs/heads/main.zip

အခန်း (၈) - Support Ticket Workflow

ဒီအခန်းမှာ နောက်ထပ် လက်တွေ့ကျတဲ့ Workflow တစ်ခုလောက် ထပ်လုပ်ကြည့်ကြပါမယ်။ လုပ်ငန်းတွေမှာ ပုံမှန်အားဖြင့် Customer က အခက်အခဲတစ်ခုခုရှိလို့ Call Center တွေဘာတွေကို ဆက်သွယ်တဲ့အခါ၊ တချို့ကိစ္စတွေက ဖုန်း Call တွေ Chat Message တွေနဲ့ ပြေလည်သွားကြပေမဲ့၊ တချို့ကိစ္စတွေက ပြေလည်သွားခြင်း မရှိကြပါဘူး။ ဒါဆိုရင် နောက်တစ်ဆင့်တက်ပြီး ဖောင်တင်တာတွေဘာတွေ လုပ်ရလေ့ရှိပါတယ်။

ဒီလိုအခြေအနေမျိုးမှာ ဖြစ်စေချင်တာက ဒီလိုပါ...

> ၁။ Customer က ဖောင်ဖြည့်ပြီး Support Request တစ်ခု ပေးပို့လိုက်ရင် Ticket ID တစ်ခုထွက်လာရပါမယ်။
>
> ၂။ Customer ကို လက်ခံရရှိကြောင်း အကြောင်းပြန်ပေးရပါမယ်။
>
> ၃။ Ticket အမျိုးအစားအလိုက် သက်ဆိုင်ရာဌာနကို Notify လုပ်ရပါမယ်။ အီးမေးလ်၊ Telegram စသည်ဖြင့် အမျိုးမျိုးဖြစ်နိုင်ပြီး ဒီနေရာမှာ **Telegram** ကို အသုံးပြုကြပါမယ်။

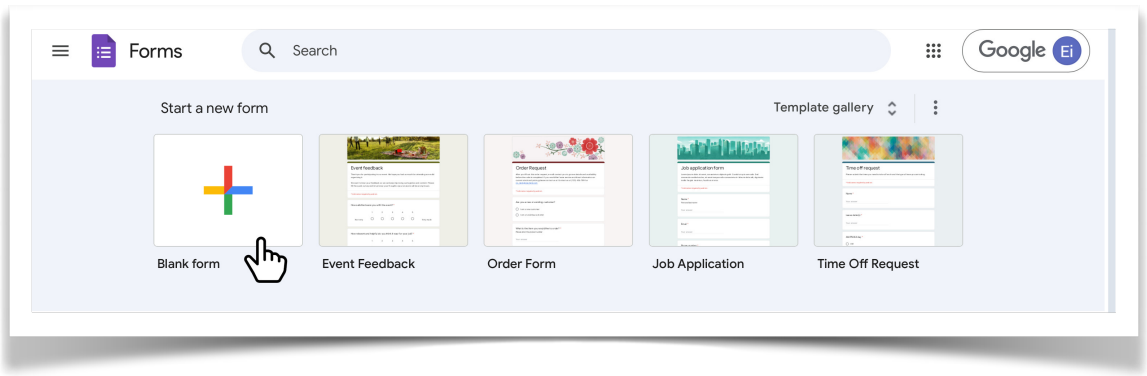
၄။ နောက်ရက်တွေမှာ Customer ရဲ့ ပြဿနာ ပြေလည်သွားရင် Follow-Up လိုက်ပြီး အဆင်ပြေရဲ့လား မေးရပါမယ်။

၅။ Customer ရဲ့ ပြဿနာက ကာလတစ်ခုကြာ မပြေလည်သေးရင် သက်ဆိုင်ရာ တာဝန်ရှိသူထံ အကြောင်းကြားရပါမယ်။

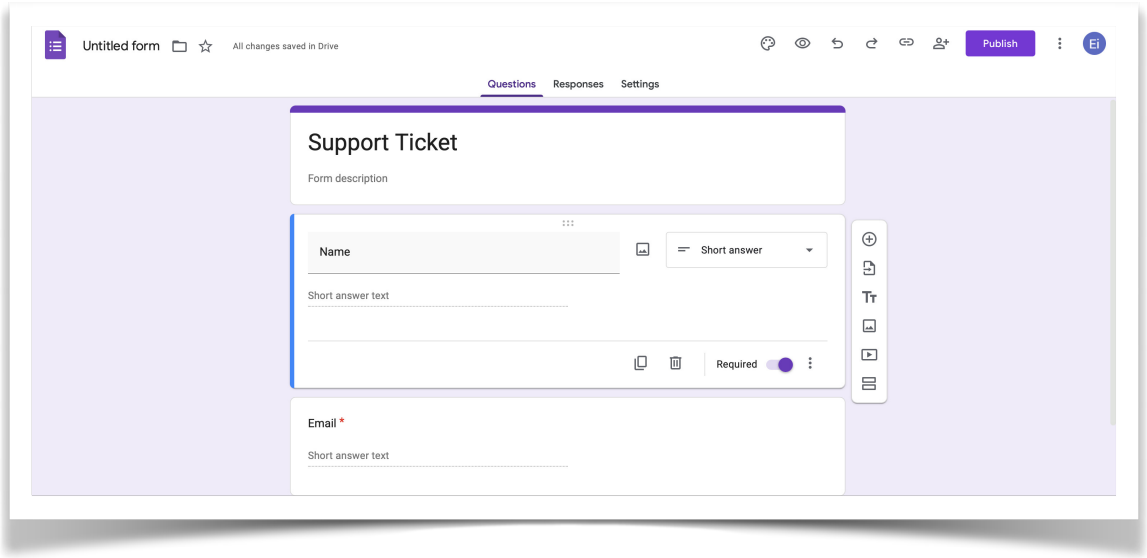
ဒီအလုပ်တွေကို လုပ်ပေးနိုင်တဲ့ Workflow ကို လုပ်ကြည့်ကြမှာပါ။ ဖောင်ဖြည့်လိုက်တဲ့ အခါ အလုပ်လုပ်စေချင်တာဖြစ်လို့၊အရင်ဆုံး **Google Form** လေးတစ်ခုလောက် ကြို လုပ်ထားဖို့ လိုပါလိမ့်မယ်။

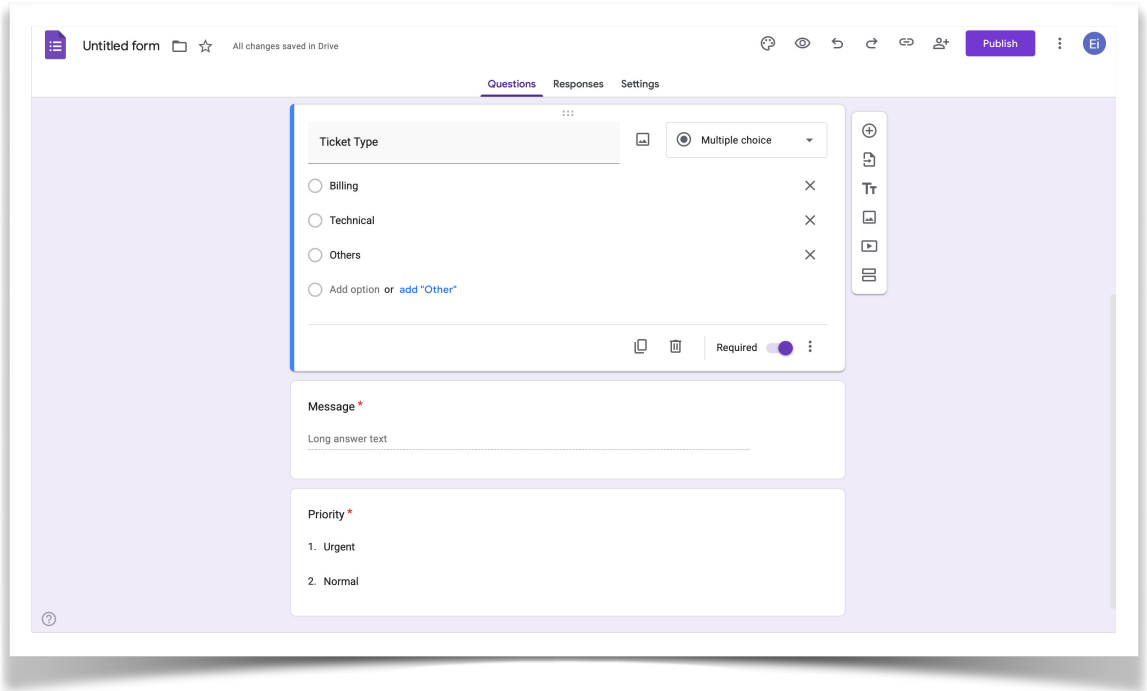
အောက်ပါလင့်ကိုသွားပြီး ဖောင်လေးတစ်ခု ဖန်တီးလိုက်ပါ။

https://docs.google.com/forms



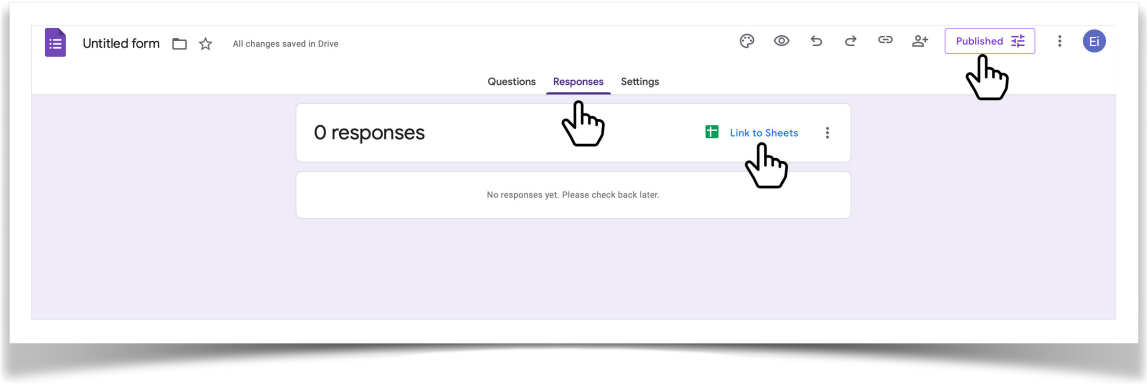
Google အကောင့်ရှိပြီးဖြစ်မယ်လို့ ယူဆပါတယ်။ အသုံးပြုရ သိပ်မခက်တဲ့အတွက် လို သလောက်ကိုပဲ ပြောချင်ပါတယ်။ ပုံနမူနာကိုကြည့်ပြီး လိုက်လုပ်လိုက်ပါ။


Untitled form
All changes saved in Drive
Publish
Ei
Questions
Responses
Settings
Support Ticket
Form description
Name
Short answer
Short answer text
Required
Email *
Short answer text

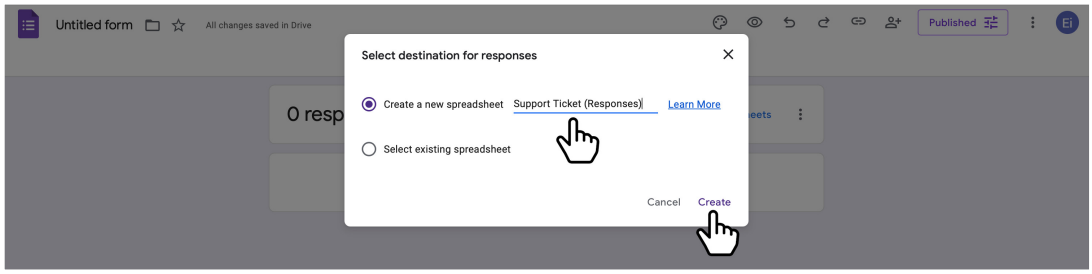

Untitled form
All changes saved in Drive
Publish
Ei
Questions
Responses
Settings
Ticket Type
Multiple choice
Billing
Technical
Others
Add option or add "Other"
Required
Message *
Long answer text
Priority *
1. Urgent
2. Normal

Form ရဲ့ အမည်ကို **Support Ticket** လို့ပေးလိုက်ပါတယ်။ ပြီးတဲ့အခါ Name, Email, Ticket Type, Message နဲ့ Priority တို့ကိုဖြည့်လို့ရတဲ့ Field တွေ ထည့်ပါတယ်။ Name နဲ့ Email အတွက် Short answer text ကို သုံးပြီး Ticket အတွက် Multiple Choice နဲ့ ရွေးစရာ **Billing**, **Technical** နဲ့ **Others** တို့ကို ထည့်ပေးထားပါတယ်။ Message အတွက် Long answer text ကို သုံးထားပါတယ်။ Priority အတွက်လည်း Multiple Choice ကိုပဲ သုံးလို့ရပါတယ်။ နမူနာမှာတော့ Dropdown ကို သုံးထားပါတယ်။

လိုအပ်တာတွေ စုံပြီဆိုရင် **Publish** လုပ်လိုက်ပါ။ ပြီးတဲ့အခါ ပုံမှာပြထားသလို **Responses** Tab ကို သွားလိုက်ပါ။ **Link to Sheets** ကို နှိပ်လိုက်ပါ။

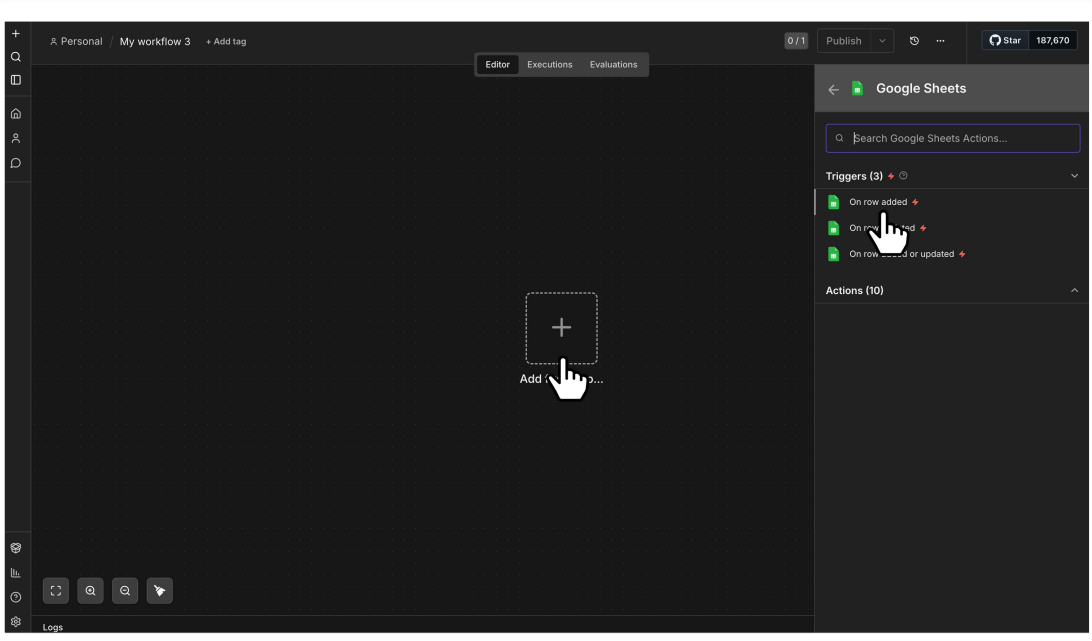


နောက်တစ်ဆင့်မှာ Sheets အမည်ကို **Support Ticket (Responses)** လို့ပေးလိုက်ပြီး Create လုပ်လိုက်ပါ။



ဒါဆိုရင် Google Form တစ်ခု ဖန်တီးပြီးသွားပြီဖြစ်သလို အဲ့ဒီဖောင်ကို Google Sheets နဲ့လည်း ချိတ်ဆက်ပြီးစီးသွားပါပြီ။

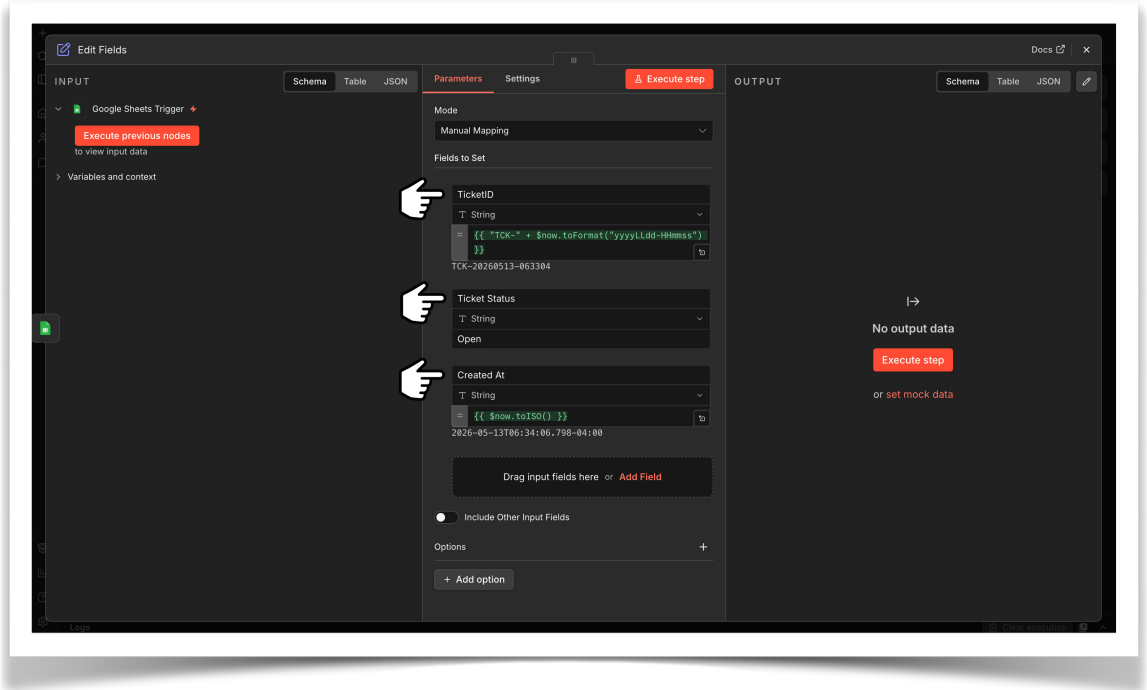
n8n နဲ့ Workflow အသစ်တစ်ခု ဖန်တီးလိုက်ပါ။ ဒီတစ်ခါ Trigger အနေနဲ့ **Google Sheets** ရဲ့ **On row add** ကို သုံးကြပါမယ်။



Sheets မှာ Row အသစ်တစ်ခု ဝင်လာတာနဲ့ ဒီ Workflow က အသက်ဝင်သွားမှာပါ။ ပြီးခဲ့တဲ့အခန်းက လုပ်ခဲ့သလိုပဲ လိုအပ်တဲ့ Credential တွေ ထည့်ပေးရပါမယ်။ Node Parameter ကို ပုံနဲ့မပြတော့ပါဘူး။ **Poll Times** ကို Every Minute ပေးထားပါလိမ့်မယ်။ ပြင်စရာမလိုပါဘူး။ တစ်မိနစ်တစ်ခါ Update ရှိမရှိ ကြည့်ပြီးအလုပ်လုပ်သွားမှာပါ။

Document အတွက် **Support Ticket (Responses)** ကို ရွေးပြီး Sheet အနေနဲ့ **Form Responses 1** ကို ရွေးထားလိုက်ပါ။

နောက်တစ်ဆင့်အနေနဲ့ **Edit Field (Set) Node** တစ်ခုထည့်ပြီး အခုလို Parameters တွေ သတ်မှတ်ပေးလိုက်ပါ။



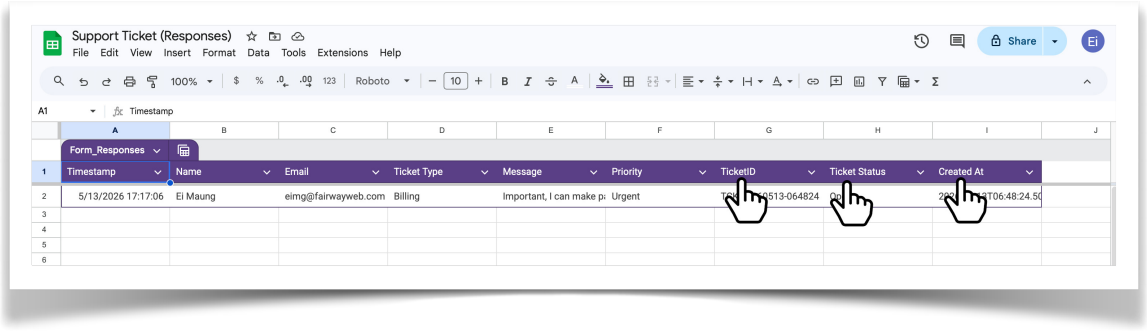
TicketID, **Ticket Status** နဲ့ **Created At** ဆိုတဲ့ Field (၃) ခု တည်ဆောက်လိုက်တာပါ။ TicketID အတွက် ဒီကုဒ်လေးထည့်ပေးရမှာပါ။

```
{{ "TCK-" + $now.toFormat("yyyyLLdd-HHmmss") }}
```

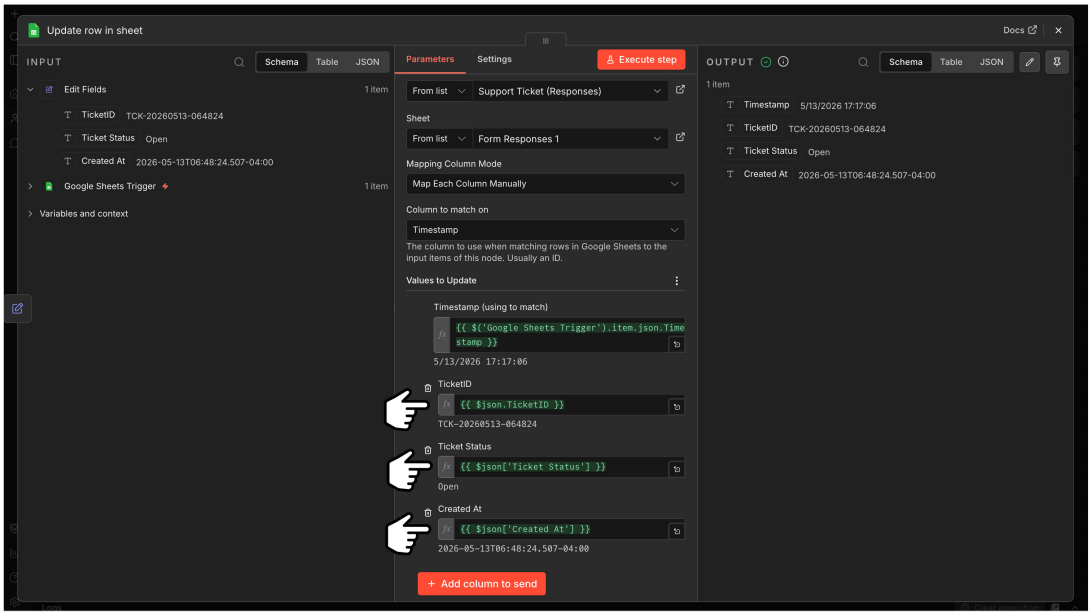
ရက်စွဲအချိန်တွေကို အခြေခံတဲ့ Format နဲ့ Ticket ID သစ် Generate လုပ်လိုက်တာပါပဲ။ **Ticket Status** ကိုတော့ **Open** လို့ပေးလိုက်ပါ။ **Created At** အတွက် လက်ရှိရက်စွဲအချိန်ကို လိုချင်လို့ အခုလို ထည့်ပေးလိုက်ပါ။

```
{{  $now.toISO()  }}
```

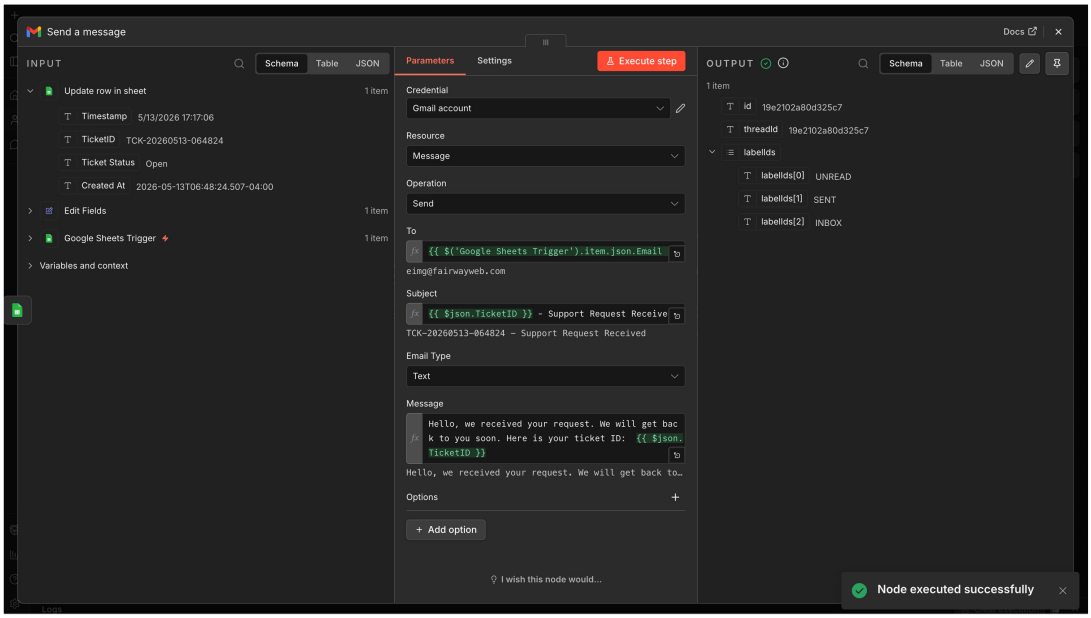
Google Form ရဲ့ **Responses** Tab ကနေ **View in Sheets** ကိုနှိပ်ပြီး **Google Sheets** မှာလည်း အခုလို အလားတူ Column တွေ ထပ်တိုးပေးလိုက်ပါ။



Workflow ကို ပြန်သွားပြီး နောက်ထပ် **Google Sheet Node** တစ်ခု ထပ်ထည့်လိုက်ပါ။ **Update row in sheets** ကိုရွေးပါ။



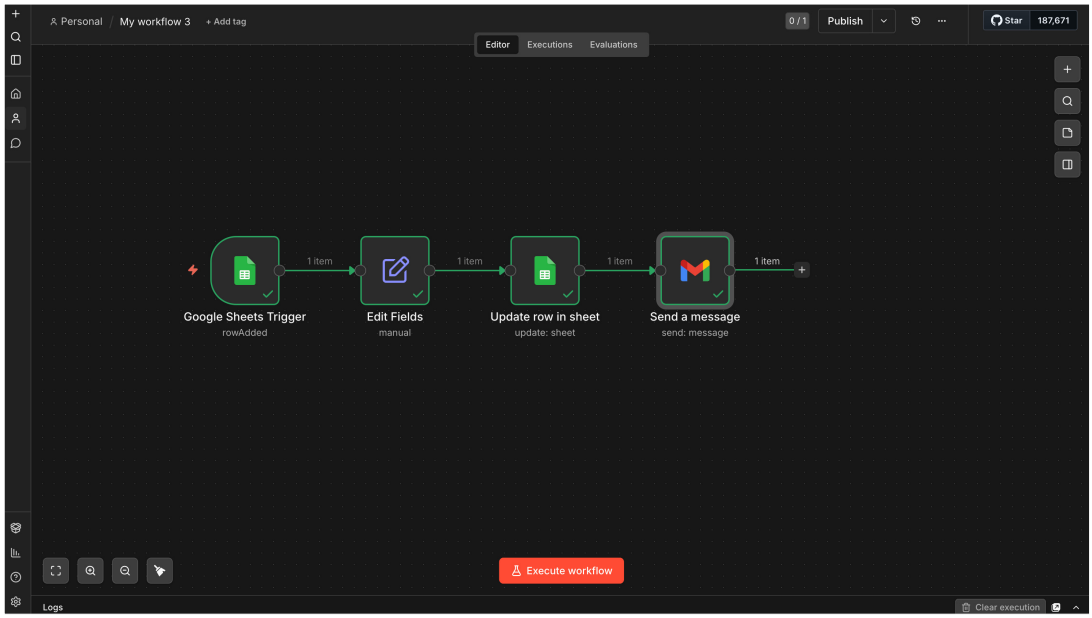
TicketID အပါအဝင် Edit Field က ရလာတဲ့ ဒေတာတွေကို Sheet ထဲမှာ မှတ်လိုက်ချင်တာပါ။ ဒါကြောင့် **TicketID**, **Ticket Status** နဲ့ **Created At** တို့ကို **Values to Update** နေရာမှာ ဆွဲထည့်ပေးလိုက်တာပါ။ နောက်တစ်ဆင့်အနေနဲ့ **Gmail Node** တစ်ခုထည့်ပြီး **Send a message** ကို ရွေးလိုက်ပါ။



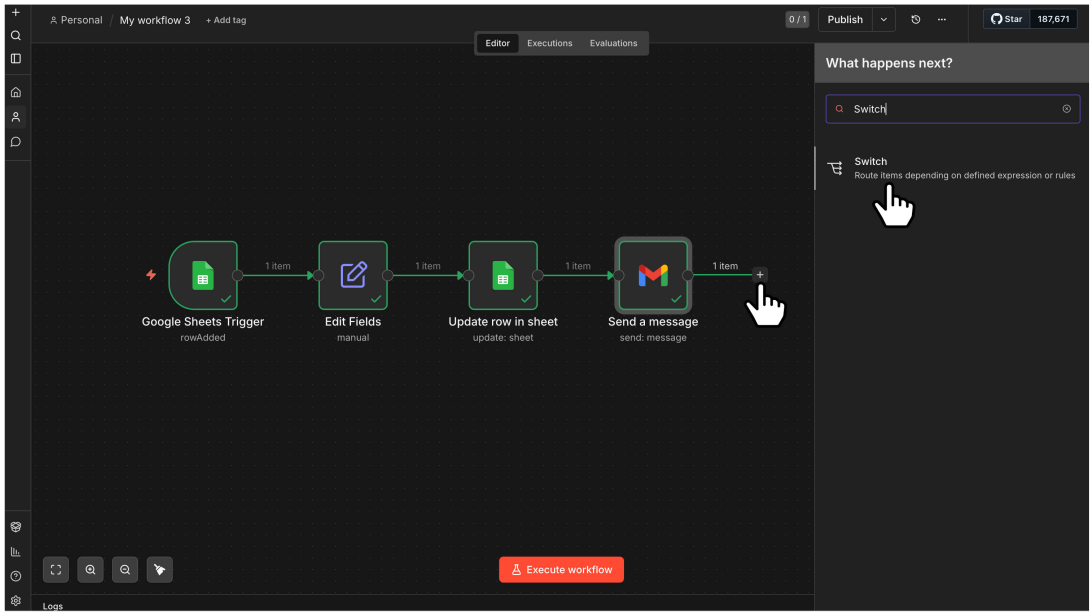
To နေရာမှာ ပေးပို့ရမဲ့ Customer ရဲ့ အီးမေးလ်လိပ်စာအဖြစ် **Google Sheets Trigger** ကနေရတဲ့ **Email** ကို ဆွဲထည့်လိုက်ပါ။ **Subject** နဲ့ **Message** ကိုတော့ နှစ်သက်ရာ ရေးဖြည့်လိုက်ပါ။ **TicketID** လေးသာ စာတစ်နေရာရာမှာ ပါပါစေ။

ဒီထိရသွားပြီဆိုရင် User က ဖောင်ဖြည့်လိုက်တာနဲ့ **TicketID** ထွက်လာပြီး၊ Customer ကို လက်ခံရရှိကြောင်း အကြောင်းကြားစာပို့တဲ့ထိ ရသွားတာပဲဖြစ်ပါတယ်။

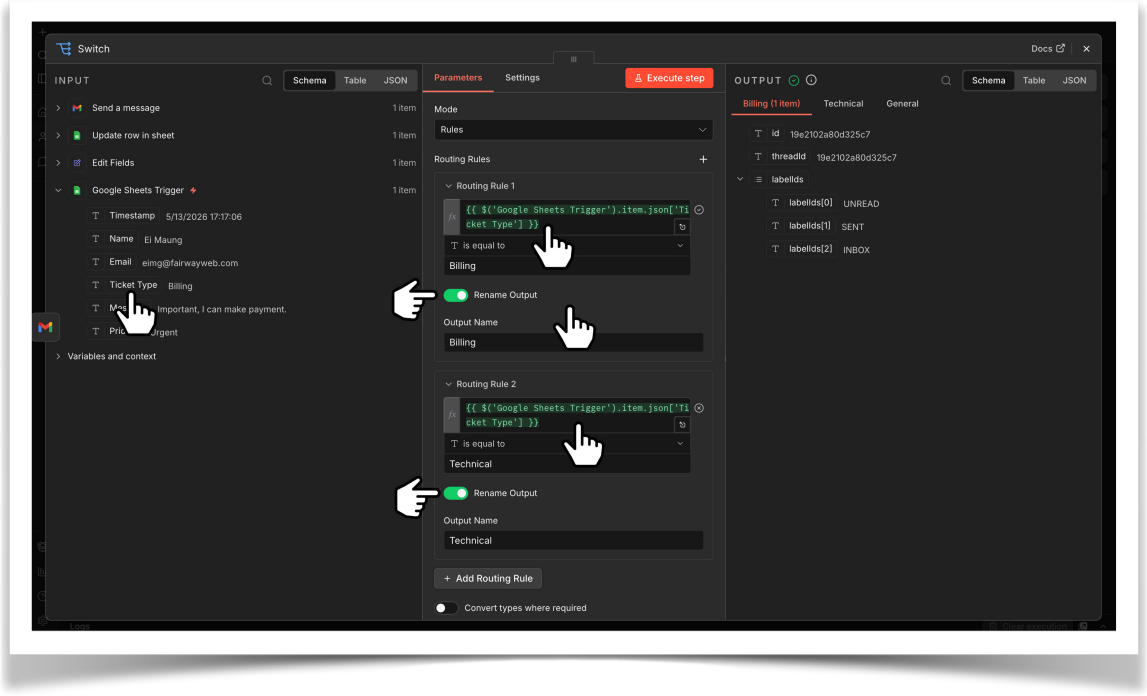
လက်ရှိ Workflow ရဲ့ ပုံစံက အခုလိုဖြစ်မှာပါ။



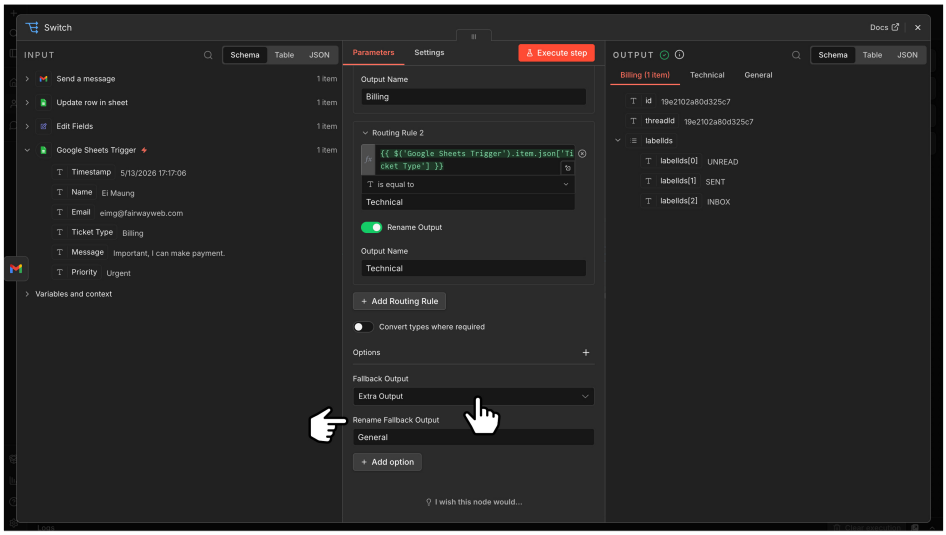
ဆက်လက်ပြီး သက်ဆိုင်ရာဌာန တာဝန်ရှိသူတွေထံ Telegram ကနေ Notify လုပ်ပါမယ်။



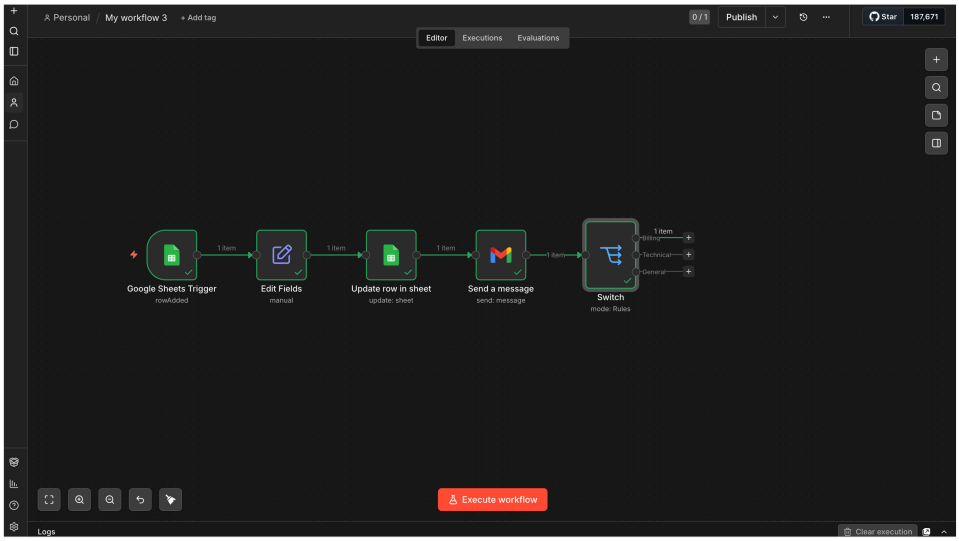
အရင်ဆုံး Ticket အမျိုးအစားပေါ်မူတည်ပြီး လမ်းကြောင်းခွဲဖို့အတွက် **Switch Node** တစ်ခု ထည့်လိုက်ပါ။ **Switch Node** ကို လမ်းကြောင်းတွေ အများကြီးထဲက သင့်တော်တဲ့ လမ်းကြောင်းကို ဆက်သွားဖို့အတွက် သုံးရပါတယ်။ **Split Node** နဲ့ မတူတာက Split Node နဲ့ ဒေတာကို ခွဲလိုက်ရင် ခွဲလိုက်သမျှ လမ်းကြောင်းတွေကို အကုန်လုပ်သွားမှာပါ။ **Switch Node** ကတော့ လမ်းကြောင်းတွေထဲက သင့်တော်ရာတစ်ခုကို ရွေးပြီး ဆက်လုပ်သွားမှာပါ။



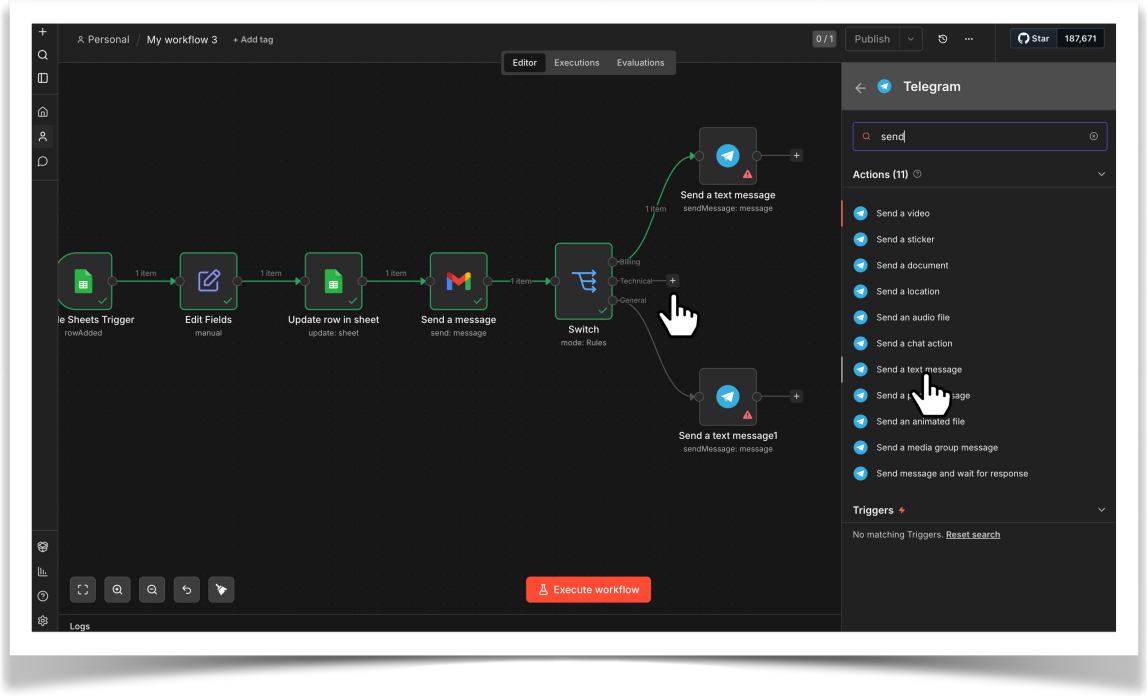
Ticket Type ကို ဆွဲထည့်လိုက်ပြီး **Ticket Type** က **Billing** ဆိုရင် လမ်းကြောင်းတစ်ခု သွားပါမယ်။ **Ticket Type** က **Technical** ဆိုရင် အခြားလမ်းကြောင်းတစ်ခုကို သွားမှာပါ။ **Rename Output** က မဖြစ်မနေ မလိုပါဘူး။ Output လမ်းကြောင်းကို သင့်တော်ရာ နာမည်ပေးထားခြင်းအားဖြင့် ကြည့်ရလွယ်စေဖို့ ဖြစ်ပါတယ်။



အပေါ်က Rule တွေ တစ်ခုမှမကိုက်ခဲ့ရင် Extra Output ဖြစ်တဲ့ **General** ဆိုတဲ့ လမ်းကြောင်းကို ဆက်သွားမှာပဲ ဖြစ်ပါတယ်။



ဒါတွေစုံအောင်ထည့်ပြီးရင် ပုံမှာပြထားသလို လမ်းကြောင်း (၃) ခုပါတဲ့ Switch Router လေးတစ်ခုကို ရရှိသွားမှာပဲ ဖြစ်ပါတယ်။ လမ်းကြောင်းသုံးခုစလုံးအတွက် **Telegram → Send a text message** လေးတွေ တွဲပေးလိုက်ပါ။

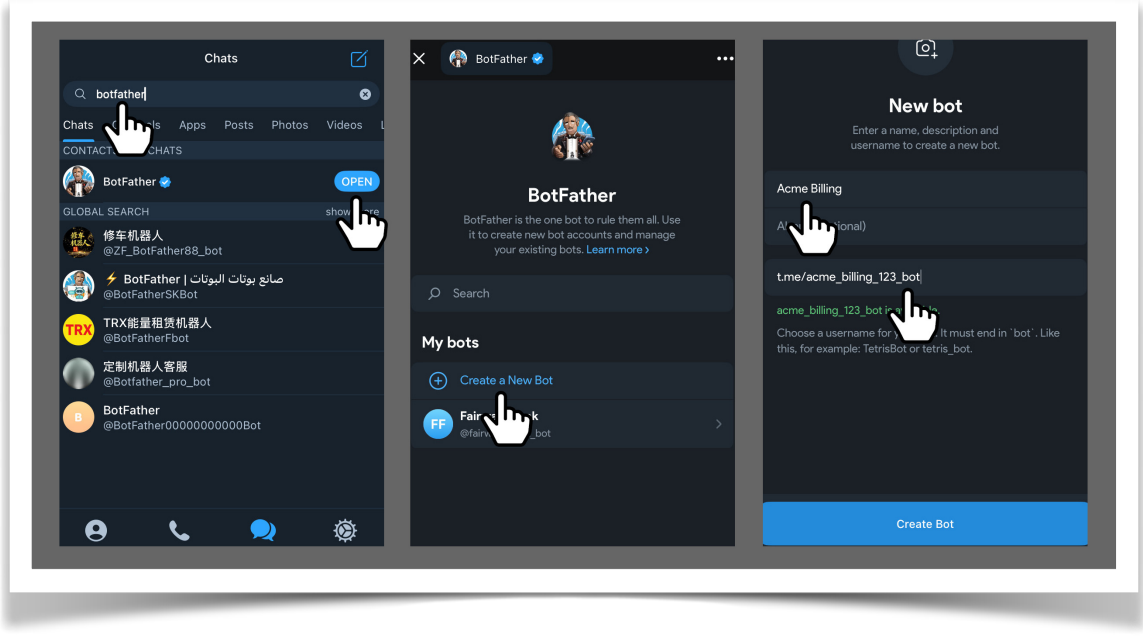


ဒီလောက်ဆိုရင် ပြည့်စုံသွားပါပြီ။ ဝင်လာတဲ့ Request မှာ Ticket Type က **Billing** ဆိုရင် Billing ဌာနရဲ့ Telegram ကို Notify လုပ်ပြီး၊ **Technical** ဆိုရင် Technical ဌာန Telegram ကို Notify လုပ်သွားမှာ ဖြစ်ပါတယ်။ အကယ်၍ တခြား Ticket Type တွေ ဖြစ်ခဲ့မယ်ဆိုရင် **General Support** ဌာနကို Notify လုပ်မှာပါ။

လက်တွေ့စမ်းသပ်နိုင်ဖို့အတွက်တော့ Telegram Credential ထည့်ပေးရပါမယ်။ ဒါကြောင့် **Telegram Bot** Create လုပ်ပြီး လိုအပ်တဲ့ အချက်အလက်တွေ ယူပုံယူနည်းကို ဆက်လက်ဖော်ပြပေးလိုက်ပါတယ်။

Telegram Access Token

ပုံမှာ ပြထားတဲ့ အဆင့်တွေအတိုင်းသာ လိုက်လုပ်သွားလိုက်ပါ။



၁။ ပထမဆုံး **@BotFather** ကို ရှာရပါတယ်။

၂။ တွေ့ပြီဆိုရင် **Open** ခလုတ်ကို နှိပ်ရပါတယ်။

၃။ နောက်တစ်ဆင့်မှာ **Create a New Bot** ကို နှိပ်ပါ။

၄။ Bot အမည်ကို နှစ်သက်ရာပေးနိုင်ပါတယ်။

၅။ Bot Handle ကိုလည်း နှစ်သက်ရာပေးနိုင်ပေမဲ့ နောက်ဆုံးက **_bot** နဲ့ အဆုံးသတ်ပေးရပါတယ်။

၆။ ပြီးသွားရင် **Create Bot** ကို နှိပ်လိုက်ပါ။ Telegram Bot တစ်ခုရသွားပါပြီ။

၇။ **Copy** ခလုတ်ကိုနှိပ်ပြီး Bot Access Token ကို ကူးယူထားလိုက်ပါ။

၈။ ကိုယ့် Bot ကို ပြန်ရှာလိုက်ပါ။

၉။ တွေ့ပြီဆိုရင် **Start** ခလုတ်နှိပ်လိုက်ပါ။

၁၀။ Hi ထားလိုက်ပါ။

ဒါဆိုရင် Access Token အသင့်ဖြစ်သွားပါပြီ။ Chat ID ကို ရှာဖို့လိုပါသေးတယ်။ Browser မှာ ဒီလိပ်စာကို ကူးထည့်လိုက်ပါ။

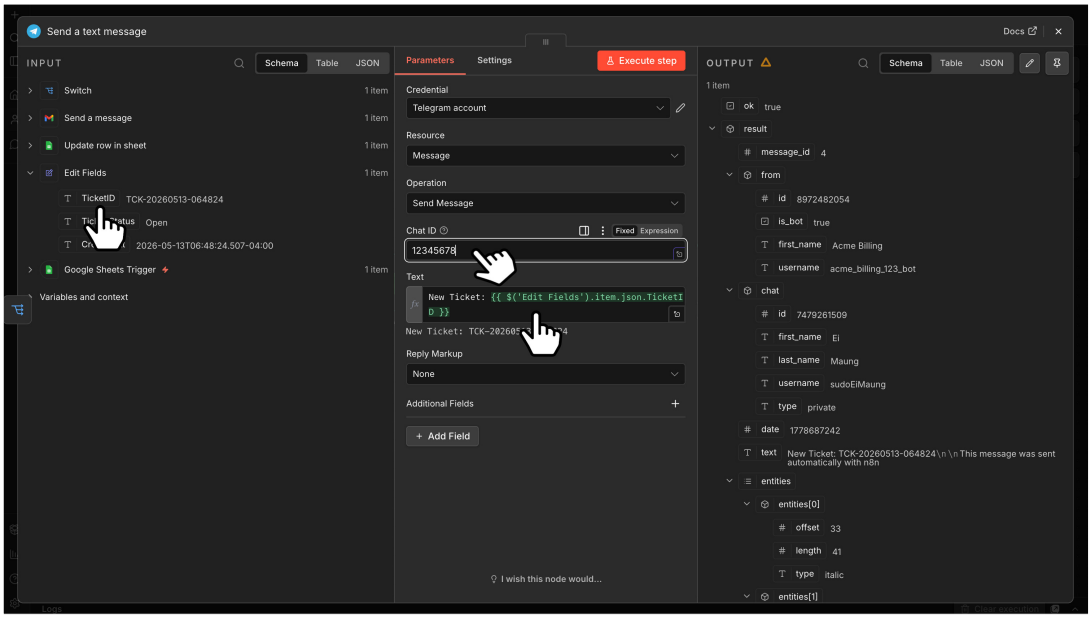
https://api.telegram.org/botYOUR_BOT_TOKEN/getUpdates

YOUR_BOT_TOKEN နေရာမှာ စောစောကကူးယူထားတဲ့ Access Token အမှန်ဖြစ်ရပါမယ်။ ဒါဆိုရင် အခုလိုအချက်အလက်တွေ ကျလာပါလိမ့်မယ်။

```
"chat": {
  "id": 123456789,
  "first_name": "Your Name",
  "type": "private"
}
```

ကျလာတဲ့ အချက်အလက်တွေထဲက **chat → id** ကို ကူးယူထားလိုက်ပါ။ အကယ်၍ အချက်အလက်တွေ၊ နမူနာပေးထားသလို ကျမလာရင် Bot ကို နောက်တစ်ကြိမ် Hi လိုက်ပါ။ ပြီးရင် နောက်တစ်ကြိမ် URL ကို ပြန်စမ်းကြည့်ပါ။

Bot **Access Token** နဲ့ **Chat ID** ရရင် n8n နဲ့ ချိတ်လို့ရပါပြီ။

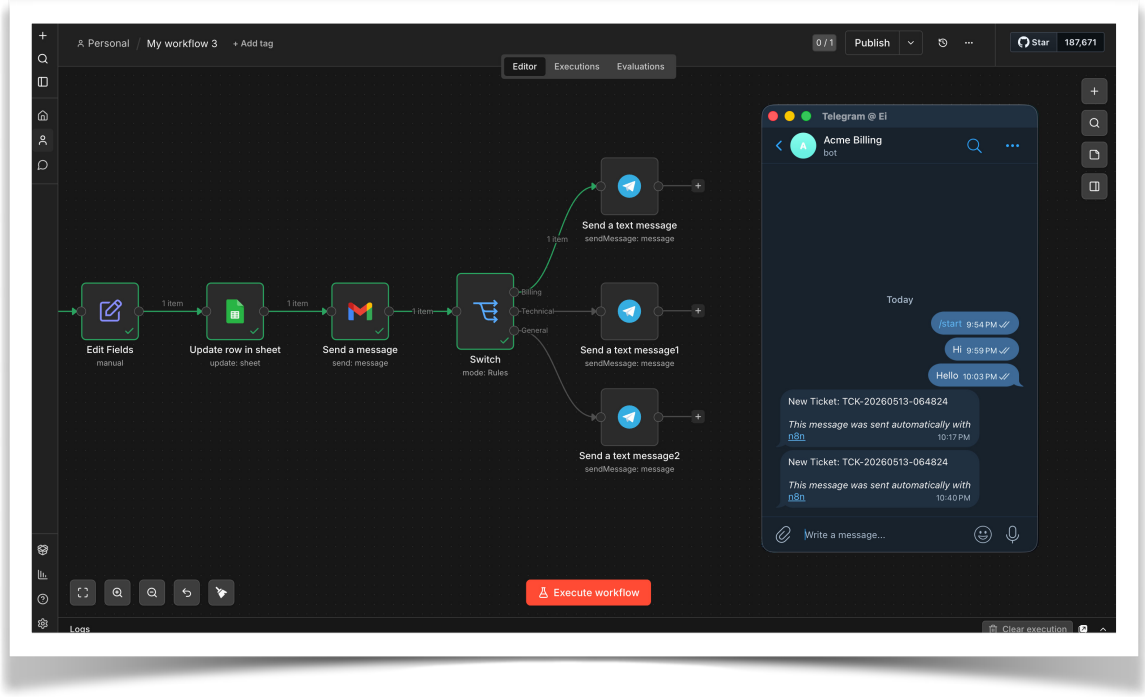


Credential ကို နှိပ်ပြီး **Access Token** ကိုထည့်ပေးလိုက်ရင် Telegram နဲ့ချိတ်မိသွားပါလိမ့်မယ်။ Telegram Node တစ်ခုချင်းစီရဲ့ **Chat ID** နေရာမှာ စောစောက ရရှိထားတဲ့ ID ကို ထည့်ပေးလိုက်ပါ။

Text နေရာမှာတော့ ပေးပို့လိုတဲ့ပုံစံနဲ့ နှစ်သက်ရာစာကို ထည့်နိုင်ပါတယ်။ နမူနာအနေနဲ့ **TicketID** ကို ဆွဲထည့်ပို့ပေးလိုက်ရင် ရပါတယ်။

ဒီနည်းနဲ့ Billing, Technical, General စသည်ဖြင့် လိုအပ်တဲ့ ဌာနအသီးသီးအတွက် Telegram Bot တွေ လုပ်ထားပေးလို့ရပါတယ်။ အကယ်၍ လူတစ်ယောက်ကို မဟုတ်ဘဲ Team တစ်ခုလုံးကို Notify လုပ်ချင်တာဆိုရင် Telegram Group ဖွဲ့ပြီး Bot ကို Group ထဲ ထည့်ထားလို့ ရပါတယ်။ ဒါဆိုရင် Notification က Group တစ်ခုလုံးဆီကို ရောက်မှာပါ။

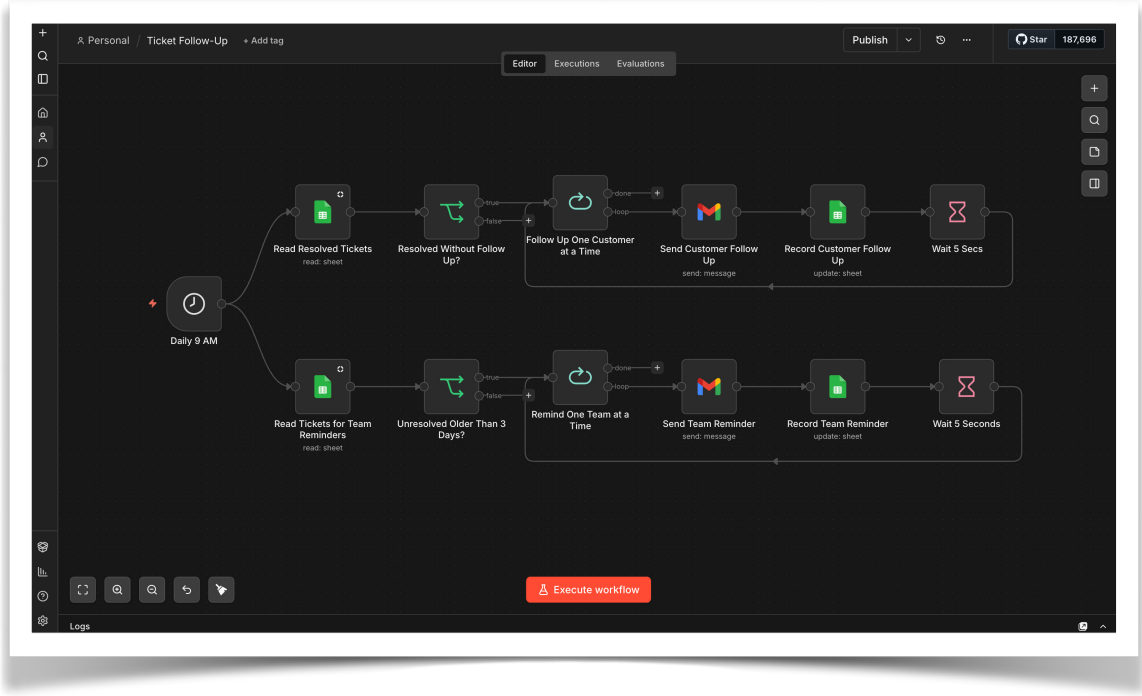
လက်ရှိနမူနာမှာတော့ Bot တစ်ခုတည်းရဲ့ Chat ID နဲ့ပဲ Telegram Node (၃) ခုလုံးကို စမ်းကြည့်လိုက်လို့လည်းရပါတယ်။



နမူနာပုံမှာ လက်တွေ့စမ်းသပ်ကြည့်ထားတဲ့ နောက်ဆုံး ရလဒ်ကို ထည့်သွင်းဖော်ပြလိုက်တာပါ။

Ticket Follow-Up Workflow

လက်စနဲ့ Ticket Follow-Up Workflow လုပ်ပုံလုပ်နည်းကိုလည်း ထည့်သွင်းဖော်ပြလိုက်ပါတယ်။ ဒါကိုတော့ တစ်ဆင့်ချင်း လုပ်မပြတော့ပါဘူး။ အိုင်ဒီယာရအောင် ဖော်ပြလိုက်တာဖြစ်ပြီး ကိုယ်တိုင် စမ်းသပ်ပြီး လုပ်ကြည့်သင့်ပါတယ်။



၁။ နေ့စဉ် မနက် (၉) နာရီမှာ လမ်းကြောင်းနှစ်ခုနဲ့ ပုံစံတူ အလုပ်နှစ်မျိုးလုပ်ပါတယ်။

၂။ ပထမဆုံးအနေနဲ့ **Google Sheets** ကနေ Ticket စာရင်းကို ရယူပါတယ်။

၃။ ပထမလမ်းကြောင်းမှာ Ticket Status က **Resolved** ဖြစ်နေပေမဲ့ Follow-Up မလိုက်ရသေးတဲ့ Customer တွေကို အီးမေးလ်ပေးပို့ပြီး Follow-Up လိုက်ပါတယ်။

၄။ Follow-Up လိုက်ပြီးကြောင်း **Google Sheets** မှာ Record လုပ်ပါတယ်။

၅။ ဒုတိယလမ်းကြောင်းမှာ (၃) ရက်ကြာတဲ့အထိ Resolved မဖြစ်သေးတဲ့ Ticket တွေကို စစ်ပြီး သက်ဆိုင်ရာ Team ကို အီးမေးလ်ပို့ Remind လုပ်ပါတယ်။

၆။ Remind လုပ်ထားကြောင်း **Google Sheets** မှာ Record လုပ်ပါတယ်။

ထည့်ချင်ရင် AI Summary တွေ ဘာတွေယူပြီး၊ တာဝန်ခံ မန်နေဂျာကို Summary ပို့တာ တွေဘာတွေ လုပ်လို့ရနိုင်ပါတယ်။ လောလောဆယ် **AI Agent Node** ကို မလေ့လာရသေးလို့ ထည့်ပြမထားပါဘူး။

နောက်တစ်ခန်းမှာ ဆက်လေ့လာကြပါမယ်။

အခန်း (၉) - AI Agent Workflow

Workflow တွေဟာ အများအားဖြင့် တိတိကျကျ ကြိုတင်သတ်မှတ်ထားတဲ့ အဆင့်တွေ များပါတယ်။ **If** တို့ **Switch** တို့လို Logic တွေ၊ **Loop** လို သဘောတရားတွေနဲ့ အခြေအနေပေါ်မူတည်ပြီး အလုပ်လုပ်နိုင်စွမ်း ရှိကြလို့ အထိုက်အလျောက် Smart ဖြစ် ပါတယ်။ လိုအပ်ရင် ကုဒ်တွေလည်း ထည့်ရေးထားလို့ ရပါသေးတယ်။

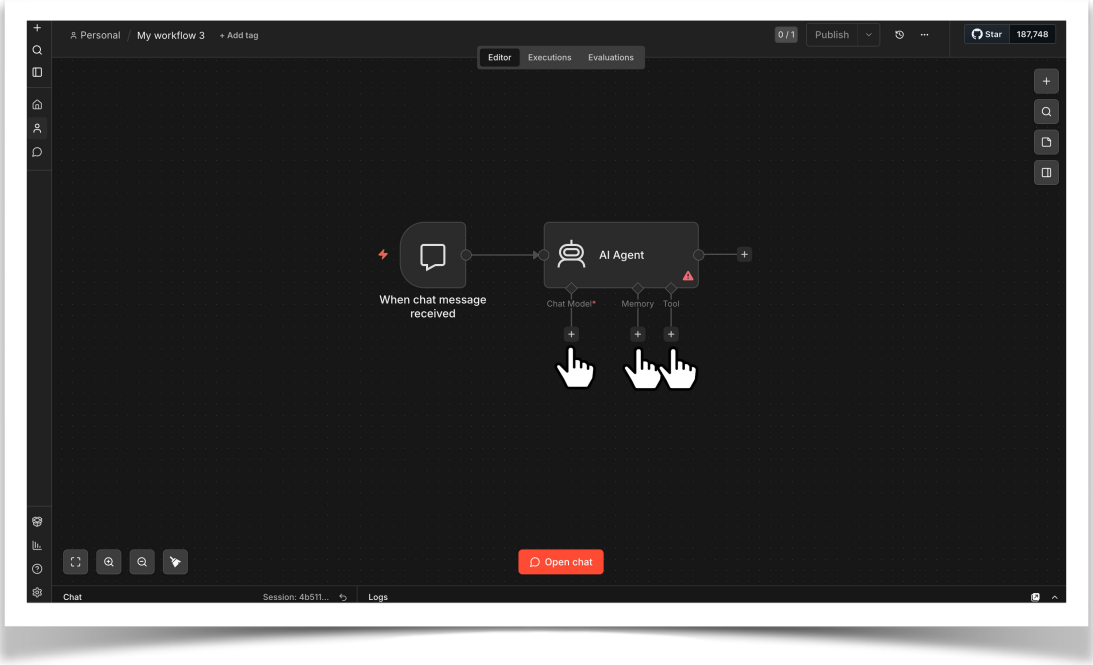
ကနေ့ခေတ်ကတော့ AI ခေတ်ဖြစ်လာတဲ့အတွက် စဉ်းစားရတာတွေ၊ ဆုံးဖြတ်ရတာတွေ AI ကို လုပ်ခိုင်းလို့လည်း ရလာပါတယ်။ **Edit Fields** တို့ **Split** တို့ **Merge** တို့နဲ့ ဒေတာကို လိုသလို ပုံစံပြောင်းရတဲ့ကိစ္စတွေကို AI နဲ့လည်း လုပ်လို့ရလာပါတယ်။

အရေးကြီးတဲ့ သဘောတရားတစ်ခု မှတ်ထားပေးပါ။ **If** တို့ **Switch** တို့ **Merge** တို့လို Workflow Node တွေက ကြိုတင်သတ်မှတ်ထားတဲ့အတိုင်း တိတိကျကျ အမြဲတမ်း အလုပ်လုပ်ကြပါတယ်။ AI Model တွေက အရမ်း Smart ဖြစ်ကြပေမဲ့ အဲ့ဒီလို အမြဲတမ်း တိကျမှာတော့ မဟုတ်ပါဘူး။ ပြန်ရတဲ့ရလဒ်တွေက၊ Input တွေ Context တွေကြောင့် ပြောင်းလဲနေမှာဖြစ်သလို အသုံးပြုထားတဲ့ Model ပေါ်မူတည်ပြီးတော့လည်း ကွဲပြားနေ မှာပါ။

ဒါကြောင့် အသုံးပြုပုံ မှန်ကန်ဖို့ လိုပါတယ်။ တိတိကျကျ အလုပ်လုပ်စေချင်တဲ့ ကိစ္စတွေအတွက် ပုံမှန် Workflow Node တွေကိုပဲ အသုံးပြုရပါတယ်။ တိကျဖို့ထက် စဉ်းစားအဖြေထုတ်ဖို့ လိုတဲ့ကိစ္စတွေကျတော့မှသာ AI Agent Node ကို အသုံးပြုရပါတယ်။

ဥပမာ - Report Summary ထုတ်တာလို ကိစ္စမျိုးဆိုရင်လည်း ကိုယ့်ဘာသာ ဒေတာတွေကို လိုတာရွေးပြီး တွက်ချက်ယူမဲ့အစား AI Agent ကို Summary ထုတ်ခိုင်းလိုက်တာက ပိုကောင်းတဲ့ ရလဒ်တွေ ရနိုင်တာ ဖြစ်နိုင်ပါတယ်။ သူ့အားသာချက်နဲ့သူ ရှိကြလို့ သူ့နေရာနဲ့သူ ကိုယ်စီ အသုံးဝင်ကြပါတယ်။

နမူနာလေးတချို့ စမ်းကြည့်လို့ရအောင် n8n Workflow အသစ်တစ်ခု ယူလိုက်ပါ။ ပြီးရင် **Chat Trigger Node** နဲ့ **AI Agent Node** တို့ကို ရှာပြီးထည့်လိုက်ပါ။ အခုလိုပုံစံ ဖြစ်ပါလိမ့်မယ်။



နမူနာအရ Chat Message ပေးလိုက်ရင် Message က AI Agent ကို ရောက်သွားမှာပါ။ AI Agent မှာ တွဲထားစရာ (၃) ခုပါတာကို တွေ့ပါလိမ့်မယ်။ **Model**, **Memory** နဲ့ **Tool** ဖြစ်ပါတယ်။

Model နေရာမှာ OpenAI, Anthropic, Google Gemini, OpenRouter စသည်ဖြင့် နှစ်သက်ရာ Provider နဲ့ Model ကို ချိတ်ဆက်အသုံးပြုနိုင်ပါတယ်။ သက်ဆိုင်ရာ Provider ရဲ့ API Key ကို Credential အနေနဲ့ ထည့်ပေးရတာပါ။

Memory ဆိုတာ Chat History ကို မှတ်ထားဖို့ပါ။ Database နည်းပညာတွေ ချိတ်ပေးလို့ ရပေမဲ့ ဒီနေရာမှာ Simple Memory ခေါ် ဘာရွပ်ထွေးမှုမှ မရှိတဲ့ နည်းပညာလေးကို သုံးလို့ ရနိုင်ပါတယ်။

Tool နေရာမှာ Google Sheets တို့ Drive တို့ Gmail တို့လို ပြင်ပ Service တွေချိတ်ထားလို့ရပါတယ်။ နောက်ထပ်တခြား AI Agent တွေ၊ HTTP Request လို လုပ်ဆောင်ချက်တွေလည်း ချိတ်ထားလို့ရပါတယ်။ အရင်လုပ်ထားတဲ့ Workflow တွေကိုတောင် ချိတ်ထားပေးလို့ ရပါသေးတယ်။

ဒီနေရာမှာ သတိပြုရမှာက Tool မှာ ချိတ်ပေးလိုက်တဲ့ Node တွေဟာ Workflow ရဲ့ အစိတ်အပိုင်း မဟုတ်ဘဲ၊ Agent က လိုအပ်ရင်ခေါ်သုံးလို့ ရအောင် ချိတ်ပေးတယ်ဆိုတဲ့ သဘောပါ။

နမူနာအနေနဲ့ Model အတွက် **OpenRouter Chat Model** ကို ချိတ်ပေးလိုက်ပါ။ Memory အတွက် **Simple Memory** ကို ချိတ်ပေးလိုက်ပါ။ Tool အတွက် **Google Sheets** ကို ချိတ်ပေးလိုက်ပါ။

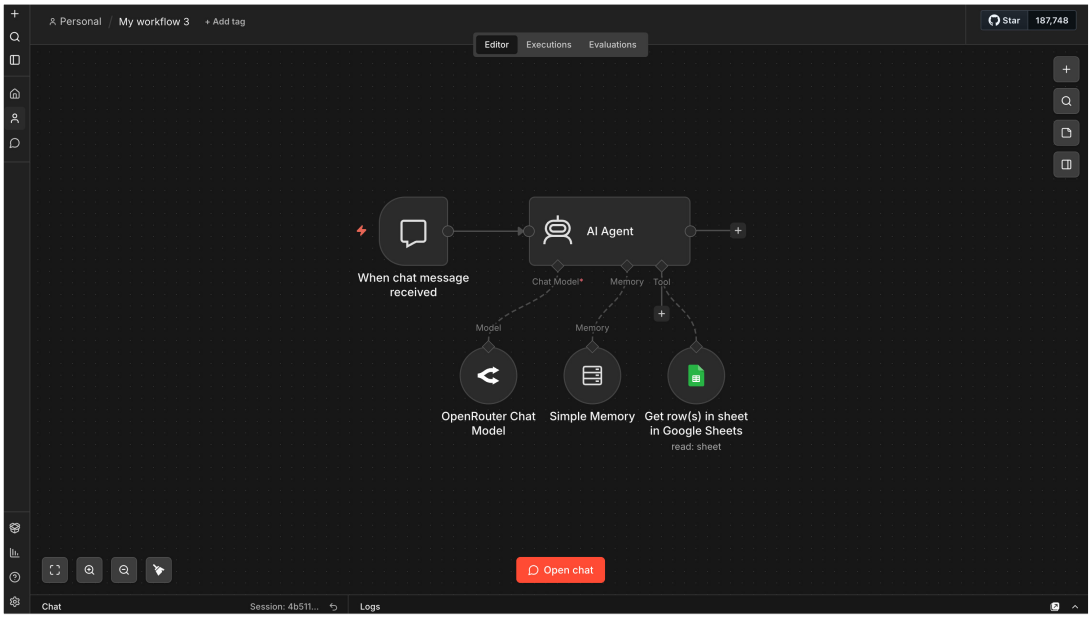
OpenRouter Credential အတွက် API Key ကို လိုအပ်ရင် ဒီလင့်မှာရယူပါ။

https://openrouter.ai/

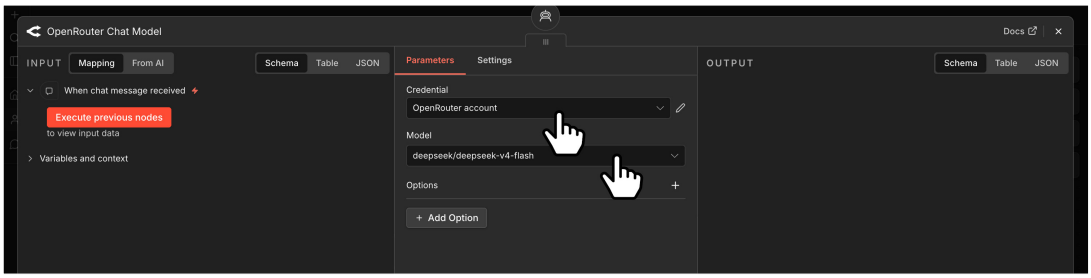
API Key ကို အခမဲ့ရယူနိုင်ပြီး အခမဲ့သုံးလို့ရတဲ့ Models တွေလည်း ရှိပါတယ်။

https://openrouter.ai/models?q=free

အခမဲ့ရလဒ် Model တွေက Limit တွေတော့ ရှိတတ်ပါတယ်။ ပိုကောင်းတဲ့ ရလဒ်တွေရချင်ရင် Credit ထည့်ထားဖို့ လိုပါတယ်။ $5 လောက် ထည့်ထားလိုက်ရင် ကောင်းကောင်းစမ်းလို့ ရနိုင်ပါတယ်။ ဒါဆိုရင် GPT တို့ Claude တို့ Gemini တို့လို ရှေ့ဆောင် Model တွေလည်း သုံးလို့ရသွားသလို၊ ဈေးသက်သာတဲ့ DeepSeek တို့ Kimi တို့ MiniMax တို့လို Model တွေကိုလည်း သုံးလို့ရသွားပါတယ်။

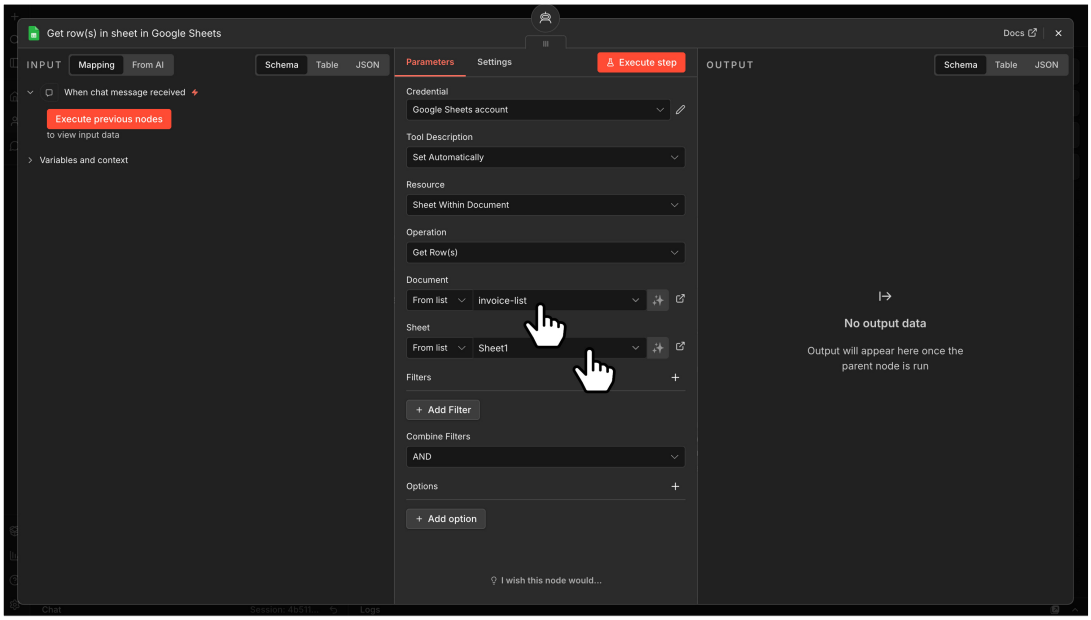


နမူနာမှာ Model အတွက် OpenRouter ရဲ့ DeepSeek v4 Flash ကို ရွေးထားပါတယ်။



Simple Memory ကိုတော့ Parameter တွေ ပြောင်းမထားပါဘူး။ **Context window length** ကို 5 လို့ သတ်မှတ်ထားတဲ့အတွက် Chat Message (၅) ခုထိ မှတ်သား အလုပ်လုပ်မယ်ဆိုတဲ့ အဓိပ္ပာယ်ဖြစ်ပါတယ်။

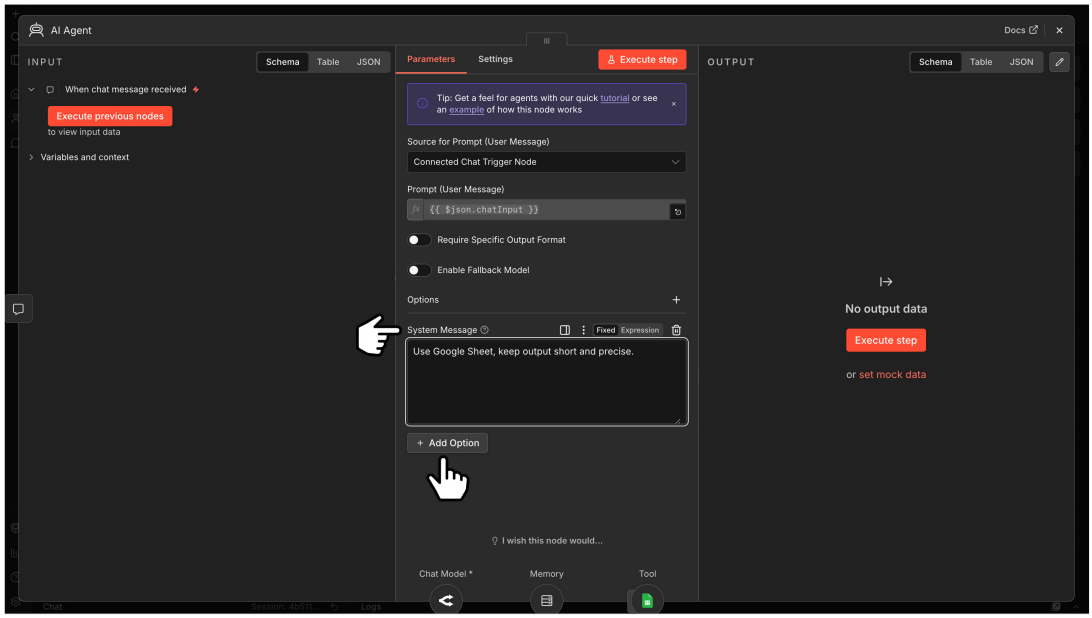
Tool အတွက် ချိတ်ပေးထားတဲ့ Google Sheet ကိုတော့ အခုလို ရွေးပေးထားပါတယ်။

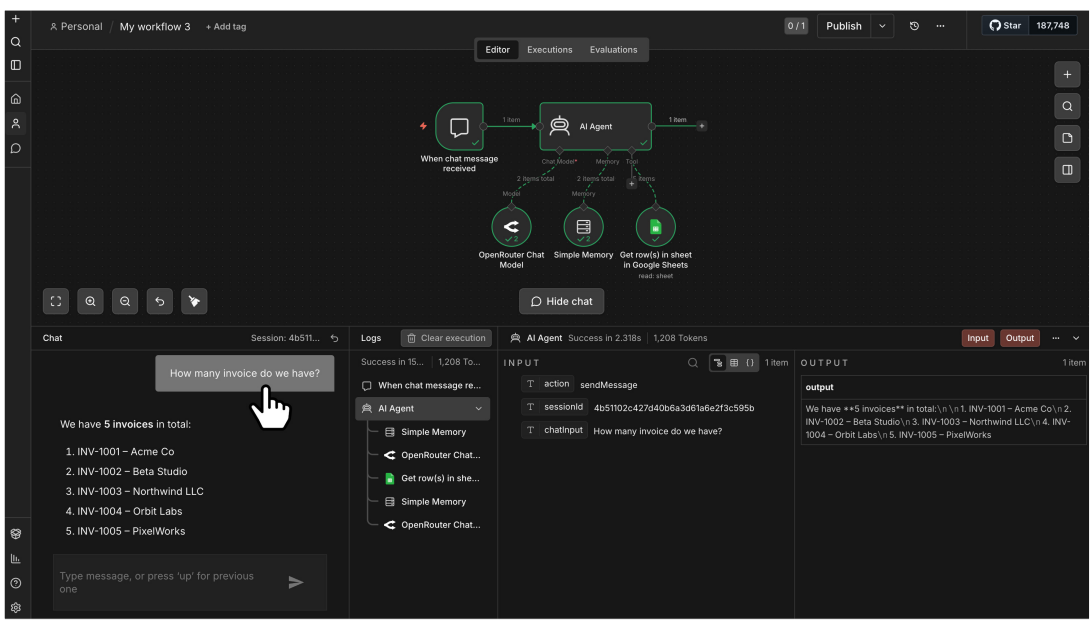


ရှေ့အခန်းတစ်ခုမှာ စမ်းလက်စဖြစ်တဲ့ **invoice-list** ကို ပြန်သုံးထားတာပါ။ AI Agent အတွက် Tool တွေ ချိတ်ဆက်တဲ့အခါ သုံးလေးငါးခု လိုသလောက် ချိတ်ဆက်နိုင်ပါတယ်။

AI Agent ကိုယ်တိုင်ကိုလည်း နောက်တစ်ဆင့်ကပုံမှာ ပြထားသလို System Prompt အပါအဝင် Parameter တွေ ပေးထားလို့ရပါတယ်။

Google Sheet သုံးပါ။ အဖြေကို တိုတိုနဲ့ တိတိကျကျပဲ ပြောပါလို့ ပြောထားပါတယ်။


AI Agent
Docs
INPUT
Schema
Table
JSON
When chat message received
Execute previous nodes
to view input data
Variables and context
Parameters
Settings
Execute step
Tip: Get a feel for agents with our quick tutorial or see an example of how this node works
Source for Prompt (User Message)
Connected Chat Trigger Node
Prompt (User Message)
{{ $json.chatInput }}
Require Specific Output Format
Enable Fallback Model
Options
System Message
Fixed
Expression
Use Google Sheet, keep output short and precise.
+ Add Option
I wish this node would...
Chat Model *
Memory
Tool
OUTPUT
Schema
Table
JSON
No output data
Execute step
or set mock data
Chat
Session: 4b511...
Logs


Personal
My workflow 3
+ Add tag
0 / 1
Publish
Star
187,748
Editor
Executions
Evaluations
When chat message received
AI Agent
1 item
Chat Model*
Memory
Tool
2 items total
2 items total
2 items
Model
Memory
OpenRouter Chat Model
Simple Memory
Get row(s) in sheet in Google Sheets
Hide chat
Chat
Session: 4b511...
How many invoice do we have?
We have 5 invoices in total:
1. INV-1001 – Acme Co
2. INV-1002 – Beta Studio
3. INV-1003 – Northwind LLC
4. INV-1004 – Orbit Labs
5. INV-1005 – PixelWorks
Type message, or press 'up' for previous one
Logs
Clear execution
Success in 15...
1,208 To...
When chat message re...
AI Agent
Simple Memory
OpenRouter Chat...
Get row(s) in she...
Simple Memory
OpenRouter Chat...
AI Agent
Success in 2.318s
1,208 Tokens
Input
Output
INPUT
1 item
action
sendMessage
sessionId
4b51102c427d40b6a3d61a6e2f3c595b
chatInput
How many invoice do we have?
OUTPUT
1 item
output
We have **5 invoices** in total:\n\n1. INV-1001 – Acme Co\n2. INV-1002 – Beta Studio\n3. INV-1003 – Northwind LLC\n4. INV-1004 – Orbit Labs\n5. INV-1005 – PixelWorks

အားလုံးပြည့်စုံပြီမို့လို့ ပြီးခဲ့တဲ့ပုံမှာပြထားသလို "Invoice ဘယ်နှစ်ခုရှိသလဲ" လို့ Chat ကို ဖွင့်ပြီး Message ပို့လိုက်တဲ့အခါ၊ AI Agent က Google Sheet ထဲက ဒေတာတွေကို ရယူပြီး (၅) ခုရှိကြောင်း ရလဒ်ပြန်ပေးသွားတာကို တွေ့ရမှာပဲ ဖြစ်ပါတယ်။

Content Idea Generation Workflow

ဒီ AI Agent Node ကို လက်တွေ့အသုံးချကြည့်ရင်း နောက်ထပ် Workflow တစ်ခုလောက် လုပ်ကြည့်ကြပါမယ်။ အပိုင်းနှစ်ပိုင်းပါပါမယ်။ ပထမတစ်ပိုင်းအနေနဲ့ နောက်ဆုံးရ နည်းပညာသတင်းတွေကို ရယူပြီး၊ ဆိုရှယ်မီဒီယာမှာ တင်ဖို့အတွက် Content ဖန်တီးပေးတဲ့ Workflow လေးပါ။

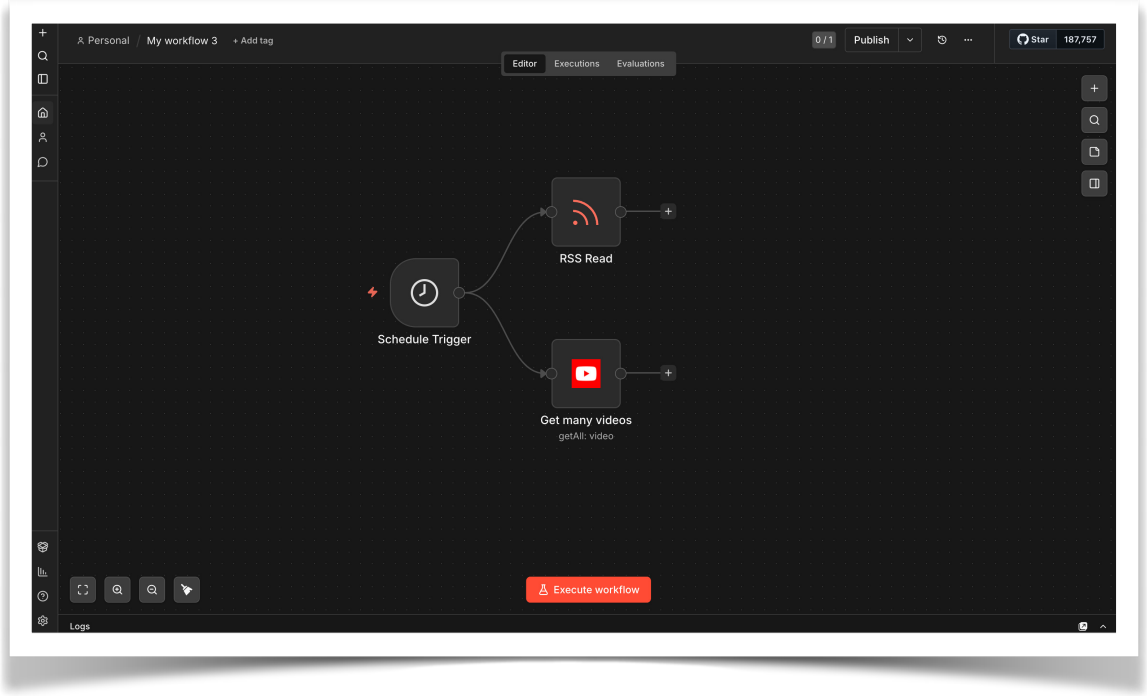
၁။ **Schedule Node** နဲ့ နေ့စဉ်မနက် (၉) နာရီ အလုပ်လုပ်ရမယ်။

၂။ **RSS Node** နဲ့ သတင်းဝဘ်ဆိုက်က သတင်းတွေယူမယ်။ နမူနာမှာ Hacker News လို့ ခေါ်တဲ့ ဝဘ်ဆိုက်ကို သုံးကြပါမယ်။

၃။ **YouTube Node** နဲ့ တစ်ရက်အတွင်းတင်ထားတဲ့ နည်းပညာဗွီဒီယိုတွေ ရယူမယ်။

၄။ အဲ့ဒီဒေတာတွေကိုပေးပြီး **AI Agent** ကို Content လေး ရေးခိုင်းမှာ ဖြစ်ပါတယ်။

ဒါတွေလုပ်ဖို့အတွက် Workflow အသစ်တစ်ခုဖန်တီးလိုက်ပါ။ **Schedule Trigger** တစ်ခု ထည့်ပြီး **RSS Node** တစ်ခုနဲ့ **YouTube Node** ရဲ့ **Get many videos** တို့ကို ထည့်လိုက်ပါ။ အခုလိုပုံစံ ဖြစ်ရပါမယ်။

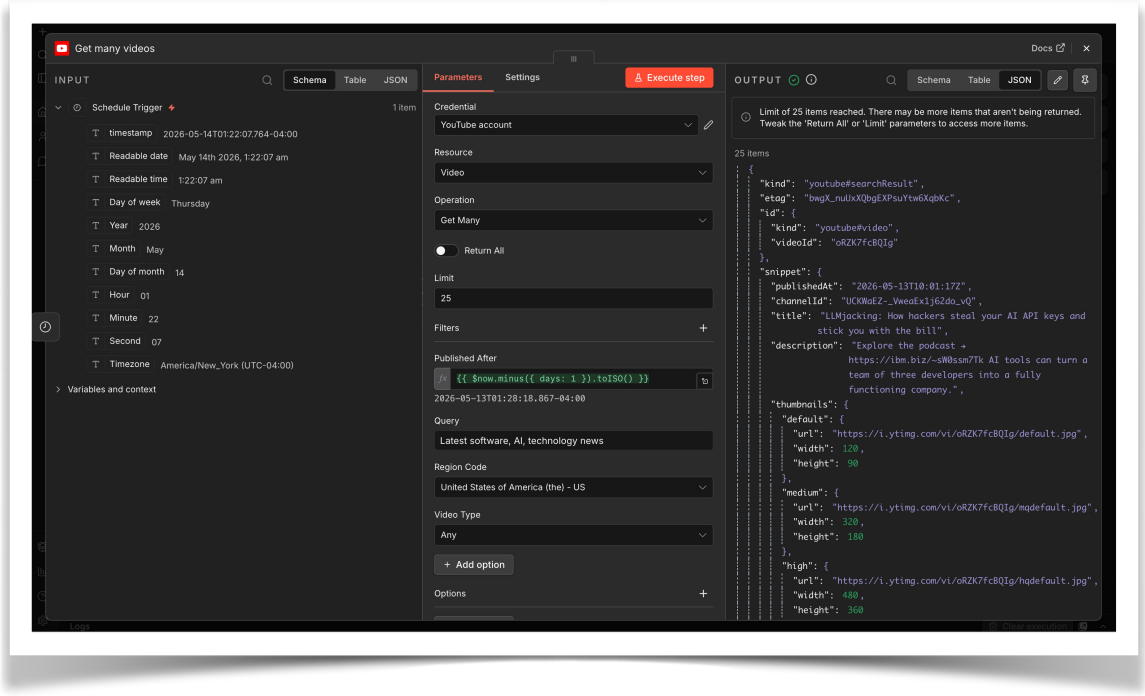


RSS Node အတွက် Parameter နမူနာကို မပြတော့ပါဘူး။ ဒီ URL လေး ထည့်ပေးလိုက်ရင် ရပါပြီ။

https://news.ycombinator.com/rss

သတင်းဝဘ်ဆိုက်တွေကနေ သတင်းတွေကို ယူတဲ့အခါ မူရင်း ဝဘ်ဆိုက်ကနေ တိုက်ရိုက်ဆွဲယူရင်လည်း ရနိုင်သလို၊ အခုလို RSS ခေါ် သတင်းဒေတာတွေချည်းပဲ သီးသန့်ပေးထားတာရှိရင်လည်း ယူလို့ရပါတယ်။ နမူနာမှာ Source အနေနဲ့ RSS တစ်ခုပဲသုံးထားပေမဲ့ လက်တွေ့အသုံးပြုချင်ရင် သတင်းဝဘ်ဆိုက်ပေါင်းများစွာကနေ ချိတ်ဆက်ရယူလို့ ရနိုင်ပါတယ်။

နောက်ထပ် Source တစ်ခုအနေနဲ့ YouTube မှာတင်ထားတဲ့ နည်းပညာဗွီဒီယိုတွေကို ရယူထားပါတယ်။ Title နဲ့ Description လောက်ပဲပါတာပါ။ တကယ်လက်တွေ့သုံးချင်ရင်တော့ Transcript ထိပါယူနိုင်ရင် ပိုပြည့်စုံမှာ ဖြစ်ပါတယ်။



YouTube အတွက်လိုအပ်မဲ့ Permission Scope ကို ပြီးခဲ့တဲ့အခန်းမှာ ကတည်းက Google Cloud Console မှာ ပေးခဲ့ပြီးဖြစ်ပါတယ်။ ဒါကြောင့် ရရှိထားတဲ့ **Client ID** တွေ **Secret** တွေကိုပဲ သုံးပြီးတော့ Credential ထည့်ပေးလိုက်ပါ။

ပြီးတဲ့အခါ **+ Add Option** ခလုတ်ကိုနှိပ်ပြီး **Published After** မှာ အခုလိုထည့်ပေးပါ။

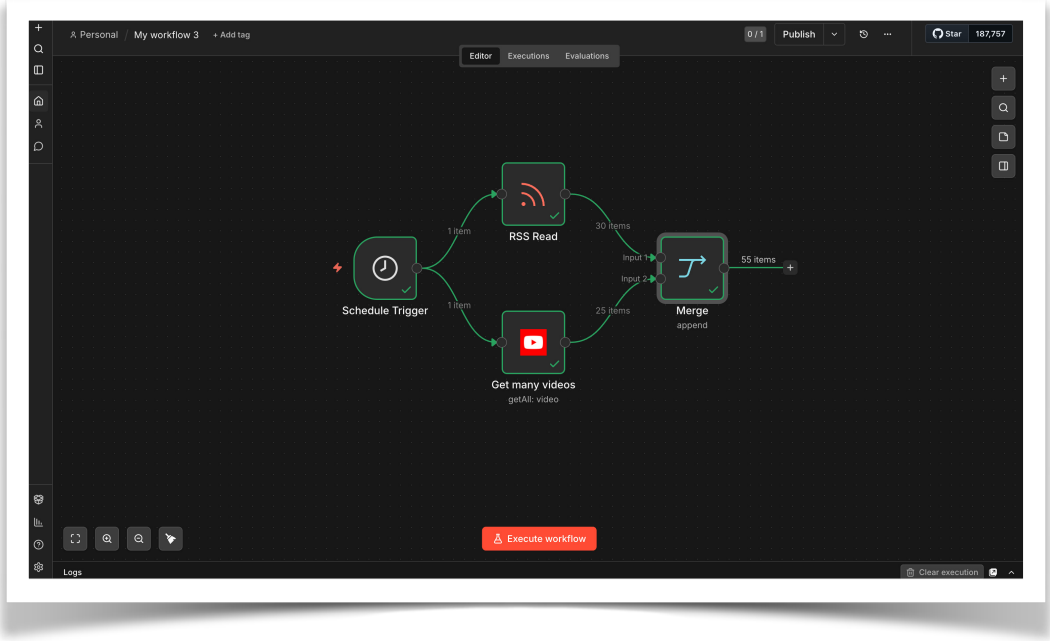
```
{{ $now.minus({ days: 1 }).toISO() }}
```

ပြီးခဲ့တဲ့ (၁) ရက်အတွင်းတင်ထားတဲ့ Update ကိုပဲလိုချင်တဲ့အတွက် ဖြစ်ပါတယ်။

ပြီးတဲ့အခါ **Query** မှာ ကိုယ်ရှာချင်ရာကို ရိုက်ထည့်လို့ရပါတယ်။ နမူနာမှာ နောက်ဆုံးရ ဆော့ဖ်ဝဲ၊ အေအိုင်နဲ့ နည်းပညာဗွီဒီယိုတွေကို ရှာယူထားပါတယ်။ **Region Code** က မ ထည့်လည်း ရနိုင်ပါတယ်။ နမူနာမှာတော့ US အခြေစိုက်ဗွီဒီယိုတွေကို လိုချင်လို့ ရွေးပေး ထားပါတယ်။ **Video Type** ကို **Any** လို့ ရွေးပေးထားပါတယ်။

ပြီးခဲ့တဲ့ နမူနာပုံရဲ့ ညာဘက်ခြမ်းမှာ စမ်းပြထားသလို၊ YouTube ကနေ (၁) ရက်အတွင်း တင်ထားတဲ့ နည်းပညာဗွီဒီယိုတွေရဲ့ ဒေတာတွေကို ပြန်လည်ရရှိမှာ ဖြစ်ပါတယ်။

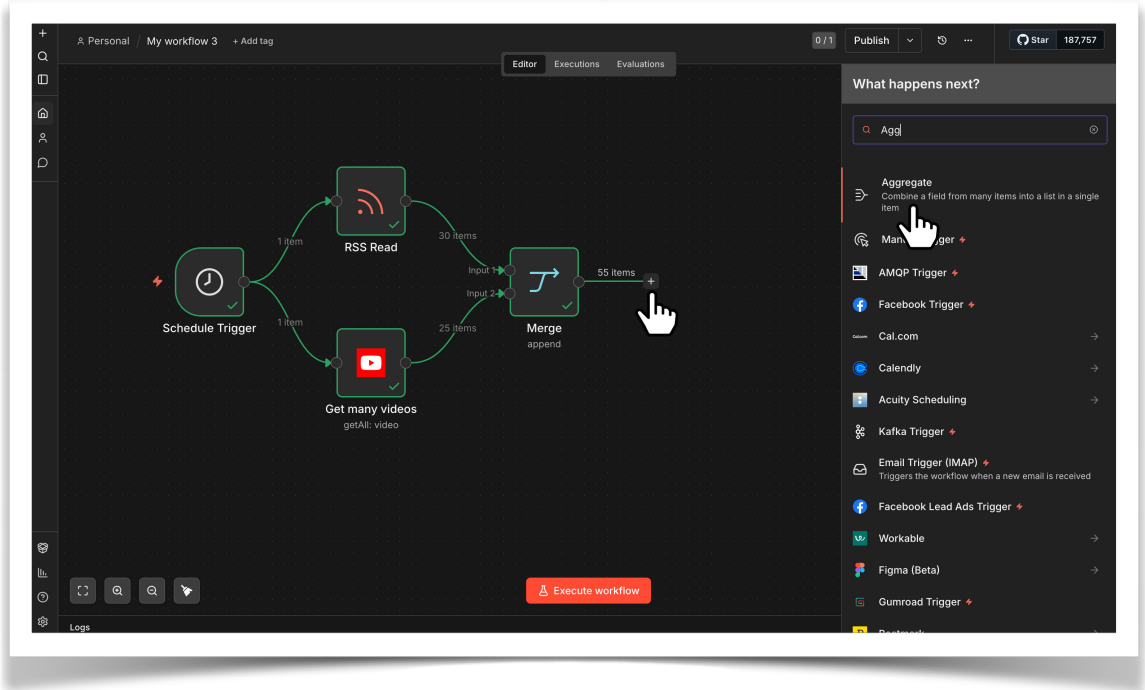
နောက်တစ်ဆင့်အနေနဲ့ **Merge Node** တစ်ခုထည့်လိုက်ပါ။ RSS ကရတဲ့ သတင်းဒေတာ တွေနဲ့ YouTube ကရတဲ့ ဒေတာတွေကို ပေါင်းစပ်လိုက်ချင်လို့ပါ။ အခုလိုပုံစံ ဖြစ်ရပါ မယ်။



Parameter တွေ ပြင်စရာမလိုပါဘူး။ စမ်းကြည့်လိုက်ရင် RSS နဲ့ YouTube ကနေပေးတဲ့ ဒေတာတွေ ပေါင်းစပ်ထားတာကို ရပါလိမ့်မယ်။ မနည်းမနောပါပဲ။

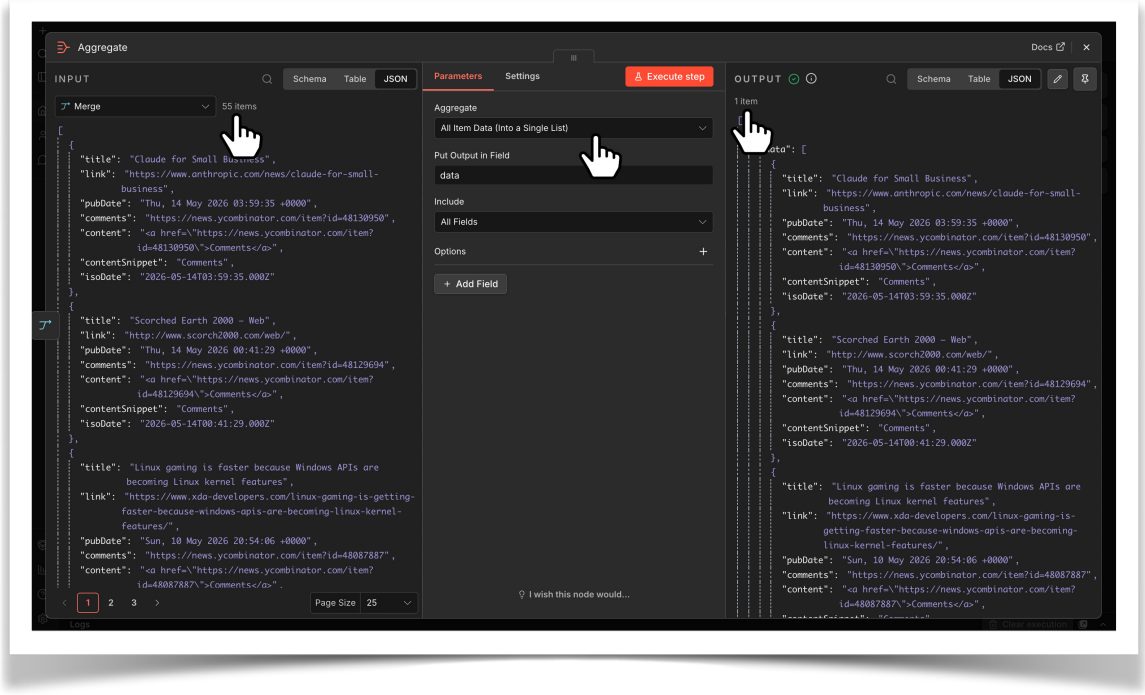
ဒီဒေတာတွေမှာ Item တွေအများကြီး ပါပါတယ်။ နမူနာအရဆိုရင် RSS ကနေ (၃၀) ခုနဲ့ YouTube ကနေ (၂၅) ခု၊ စုစုပေါင်း (၅၅) ခုထိပါပါတယ်။ နောက်ထပ် Node တွေကို ချိတ်လိုက်ရင် Item တစ်ခုကို တစ်ကြိမ်နှုန်းနဲ့ (၅၅) ကြိမ် လုပ်သွားမှာပါ။

ဒါကြောင့် ဒါတွေကို ပေါင်းပေးဖို့ လိုပါတယ်။ Connection တွေကို ပေါင်းတာ မဟုတ်တော့ပါဘူး။ ဒေတာတွေကို ပေါင်းတာပါ။ ဒါကြောင့် Merge Node နဲ့ မလုံလောက်တော့ဘဲ **Aggregate Node** ကို သုံးရပါတယ်။ ဒီလိုပါ။

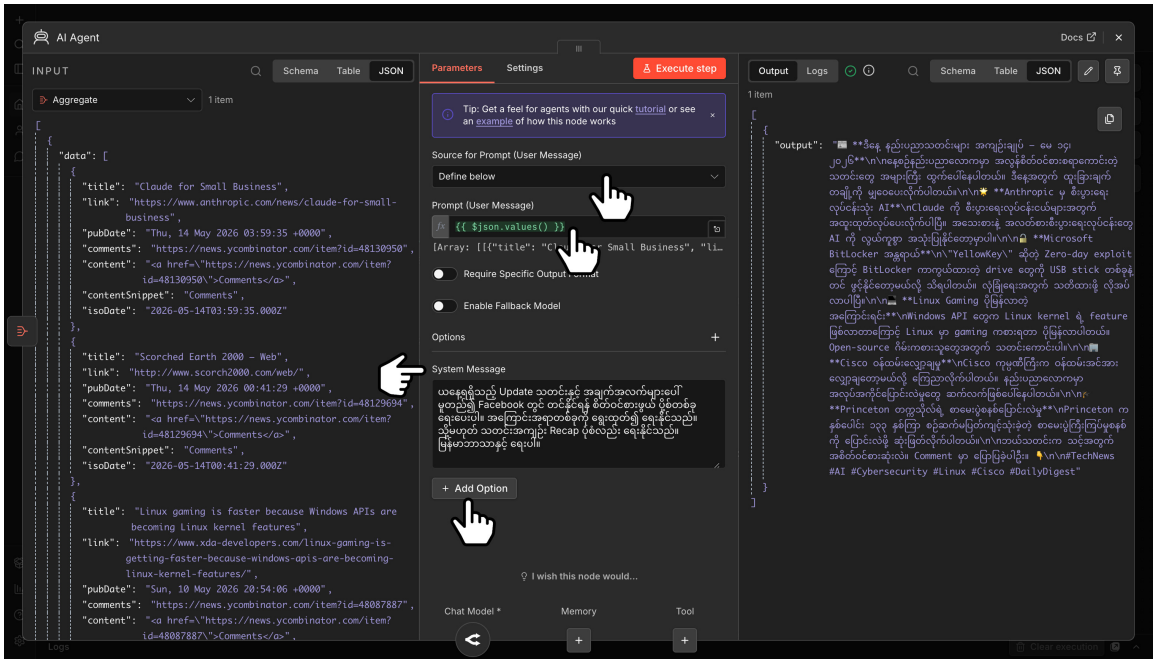


Aggregate Node အတွက် All Item Data (Into a Single List) ကို ရွေးပေးပါ။

ပုံမှာပြထားသလို မူလ 55 Items ရှိရာကနေ 1 Items ဖြစ်အောင် ပေါင်းပေးသွားပါလိမ့်မယ်။ နောက်တစ်ဆင့်မှာ **AI Agent Node** ကို ချိတ်ပေးလိုက်ပါ။



Memory တွေ Tool တွေ တစ်ခုမှ မလိုအပ်ပါဘူး။ Model ပဲလိုပါတယ်။ ဒီနမူနာမှာလည်း Model ကို **OpenRouter** ကနေရတဲ့ **DeepSeek v4 Flash** ကိုပဲ သုံးပါတယ်။



အထက်က နမူနာပုံမှာ ပြထားသလို **Source for Prompt** အတွက် **Define below** ကို ရွေးပြီး {{ $json.values() }} လို့ ထည့်ပေးလိုက်ပါ။ မူလက Chat Message ကို Prompt အနေနဲ့ ယူမှာပါ။ အခု Chat Message ကို Prompt အနေနဲ့ မသုံးဘဲ ပြီးခဲ့တဲ့ Node ကနေရတဲ့ ဒေတာတွေကို Prompt အနေနဲ့ သုံးခိုင်းလိုက်တာပါ။

System Prompt အနေနဲ့ မြန်မာလို ပို့စ်တစ်ခုရေးပေးဖို့ ပြောထားပါတယ်။ မိမိနှစ်သက်ရာ Prompt Instruction ကိုပေးထားနိုင်ပါတယ်။ နမူနာပုံရဲ့ ညာဘက်ခြမ်းမှာ DeepSeek က ရေးပေးတဲ့ မြန်မာဘာသာပို့စ်တစ်ခုကို တွေ့မြင်ရမှာပဲဖြစ်ပါတယ်။

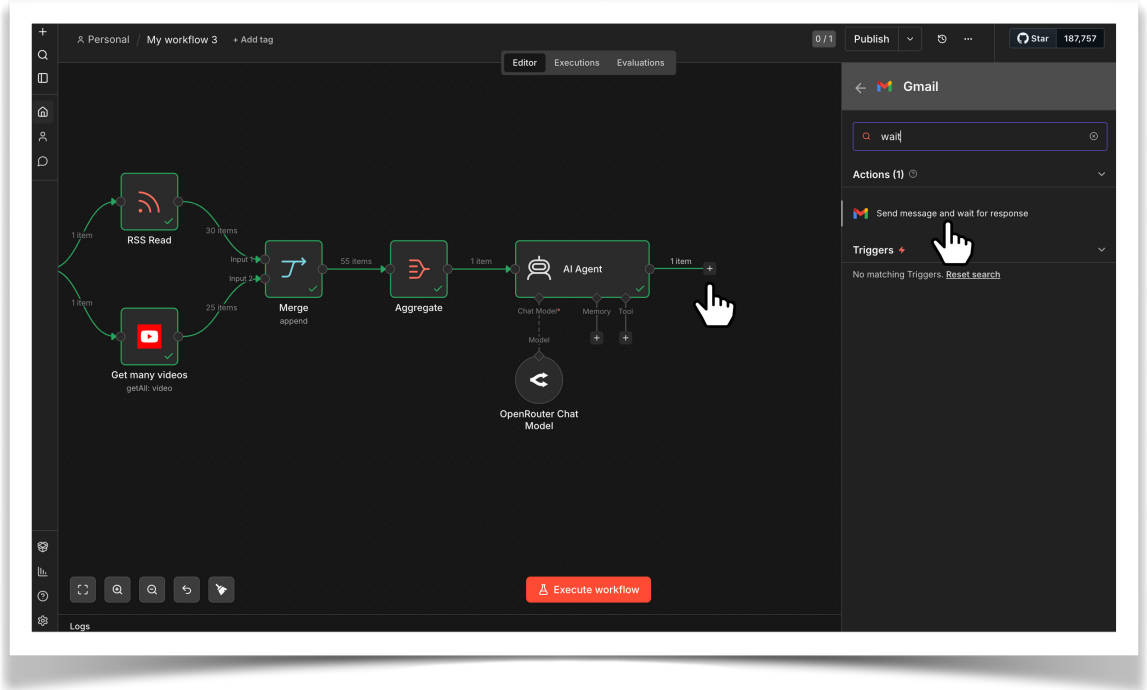
ဒါ ဟာ နော က်ဆုံး ရ သ တ င်း နဲ့ အ ချ က်အ လ က်တွေ ပေါ်မူ တ ည်ပြီး ပို့ စ်တ စ်ခု အလိုအလျောက် ရေးပေးတဲ့ Workflow ကို ရရှိလိုက်တာပါ။

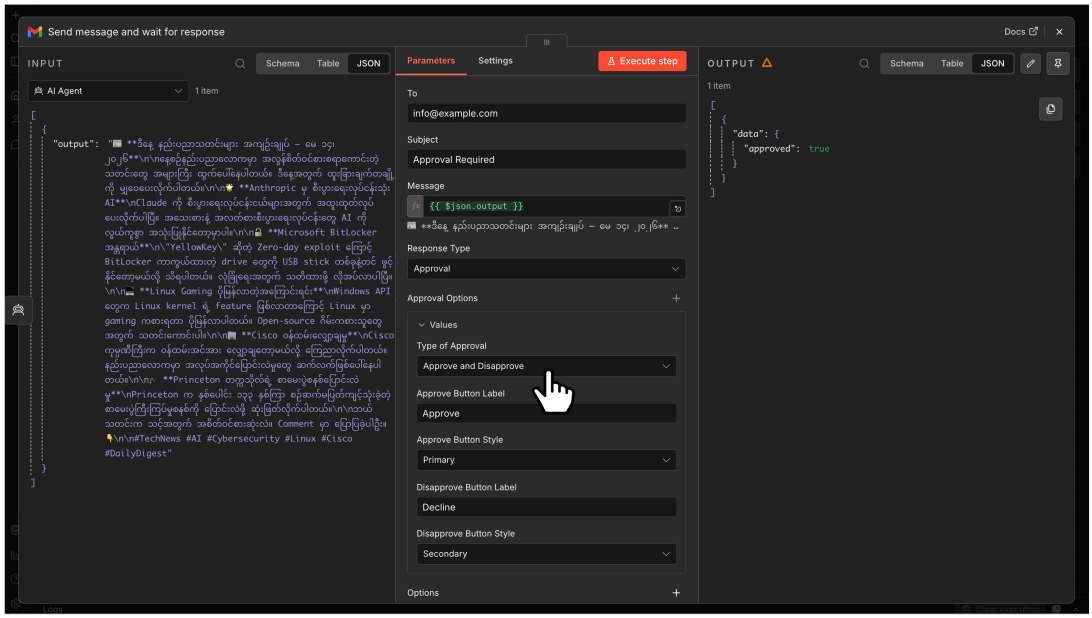
နောက်တစ်ဆင့်

နောက်တစ်ဆင့်အနေနဲ့ Content ကို ရေးပြီးတဲ့အခါ ကိုယ့်ဆီကို Email ကနေ အတည်ပြု ချက်လှမ်းတောင်းပြီး၊ ကိုယ်အတည်ပြုလိုက်တာနဲ့ ဖေ့စ်ဘုတ်မှာ အလိုအလျောက် တင် ခိုင်းလိုက်ချင်ပါတယ်။

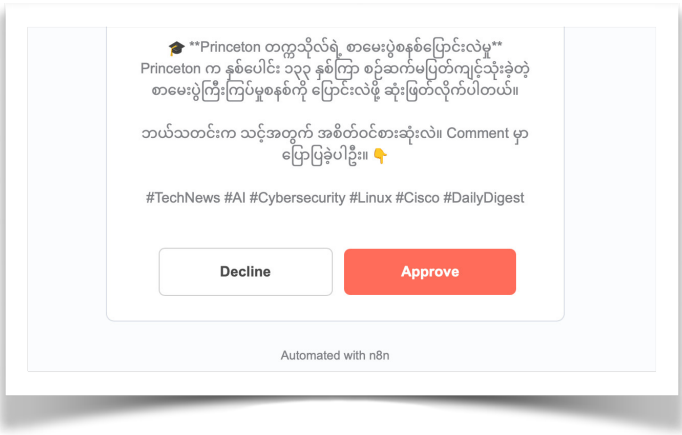
ဒီနေရာမှာ Telegram ကိုသုံးရင်လည်း ရပါတယ်။ ဒါပေမဲ့ Telegram က localhost နဲ့ အဆင်မပြေတဲ့အတွက် နောက်မှ အဲ့ဒါကို သပ်သပ် ထပ်ပြောပြပါမယ်။

Gmail Node ရဲ့ **Send message and wait for responses** ကိုပဲ ထည့်ပေးလိုက်ပါ။





To နေရာမှာ ကိုယ့်အီးမေးလ်လိပ်စာအမှန် ထည့်ပေးဖို့ လိုပါတယ်။ Message နေရာမှာ လက်ရှိရထားတဲ့ Post Output ကို ဆွဲထည့်ပေးလိုက်ပါ။ **Approve** / **Reject** ခလုတ်နှစ်ခု ပါစေချင်တဲ့အတွက် **Approve and Disapprove** ကို ရွေးထားပါတယ်။ ဒါတွေစုံအောင် လုပ်ပြီးနောက် စမ်းကြည့်လိုက်ရင် အခုလို အီးမေးလ်ဝင်လာတာကို တွေ့ရပါလိမ့်မယ်။



ကိုယ်က **Approve** ခလုတ်ကို နှိပ်ပေးလိုက်မှ Workflow က ရှေ့ဆက်သွားအောင် လုပ်ထားလို့ ရသွားပါပြီ။ ဒီနည်းနဲ့ ကိုယ့်ခွင့်ပြုချက်ရမှ လုပ်စေချင်တဲ့ အလုပ်တွေကို Workflow ထဲမှာထည့် စီစဉ်ထားလို့ရတာဖြစ်ပါတယ်။

အခုဆိုရင် အားလုံးအသင့်ဖြစ်နေပြီမို့လို့ ဖေ့စ်ဘုတ်ကို လှမ်းတင်ခိုင်းလိုက်လို့ ရနေပါပြီ။ ဒီလိုတင်လို့ရဖို့အတွက် Credential တွေ ကြိုတင်ပြင်ဆင်ထားရပါမယ်။ ဒါကြောင့် ဘယ်လိုလုပ်ရလဲ ပြောပြပါမယ်။ လိုအပ်ချက်တွေက ဒီလိုပါ။

၁။ Facebook အကောင့် ရှိရပါမယ်။

၂။ **Facebook Page** ရှိရပါမယ်။

၃။ အကောင့်က **Page Admin** (သို့) ခွင့်ပြုချက်ရှိသူ ဖြစ်ရပါမယ်။

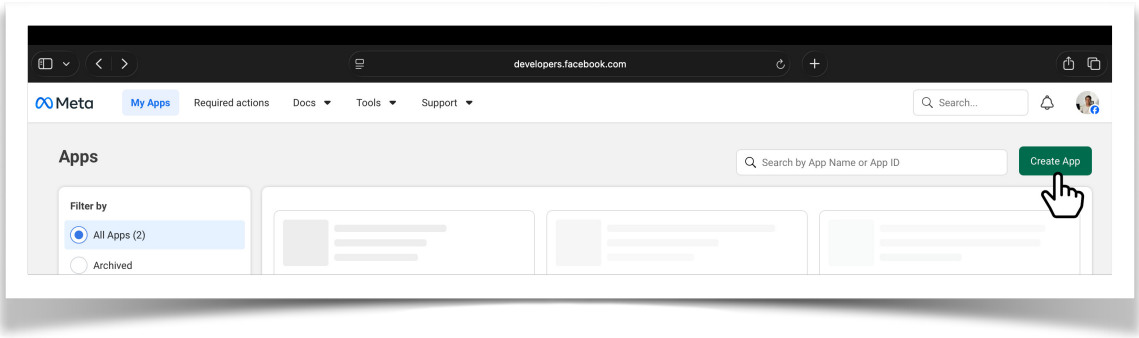
၄။ **Facebook App** ရှိရပါမယ်။

၅။ Meta Graph API အကောင့် ရှိရပါမယ်။

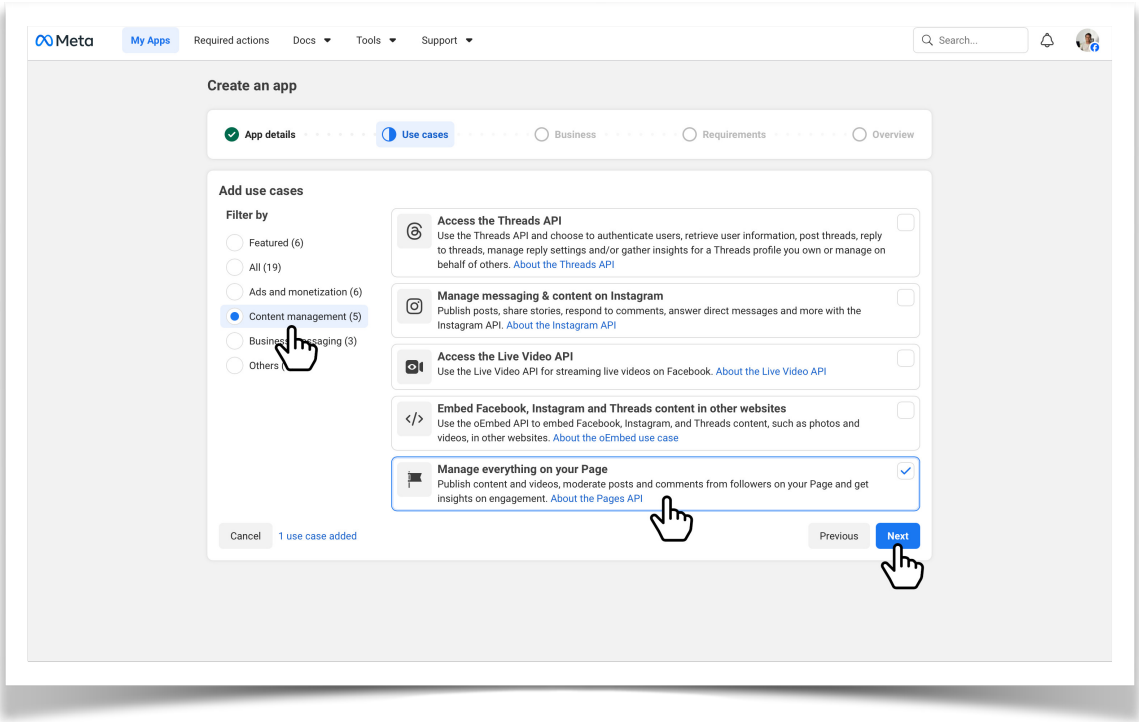
Facebook အကောင့်နဲ့ Page တော့ ရှိပြီးဖြစ်မယ်လို့ ယူဆပါတယ်။ Facebook App ဖန်တီးဖို့ ဒီလင့်ကို သွားလိုက်ပါ။

https://developers.facebook.com/apps/

ကိုယ့် Facebook အကောင့်နဲ့ပဲ Login ဝင်ထားလိုက်ပါ။ **Create App** ခလုတ်ကိုနှိပ်ပြီး App အသစ်တစ်ခု ဖန်တီးလိုက်ပါ။



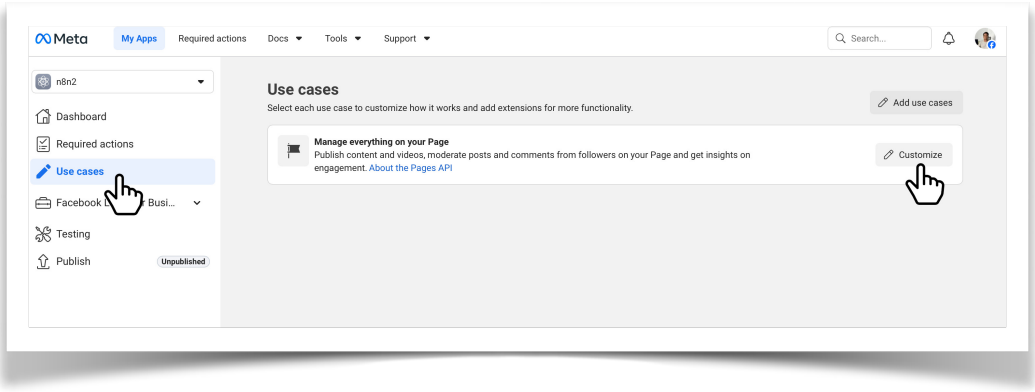
နောက်တစ်ဆင့်မှာ **App Name** နဲ့ **Email** ရေးဖြည့်ပြီး **Next** ကို နှိပ်လိုက်ပါ။



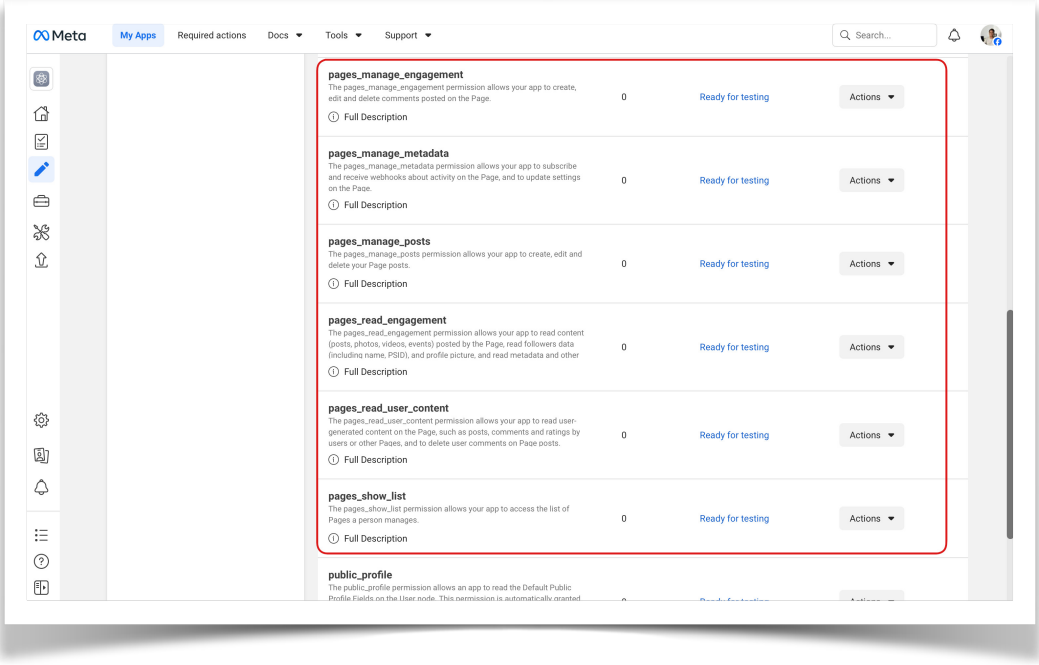
Content management ထဲက **Manage everything on your page** ကို ရွေးပြီး **Next** နဲ့ နောက်တစ်ဆင့်ကို သွားလိုက်ပါ။ နောက်တစ်ဆင့်မှာ ကိုယ့်ရဲ့ Page ကို ရွေးပြီး **Next**

ကို နှိပ်လိုက်ပါ။ နောက်အဆင့်တွေမှာ ဘာမှထပ်ရွေးစရာ မလိုတော့ **Next** နှိပ်သွားပြီး နောက်ဆုံးမှာ **Create App** ကိုနှိပ်လိုက်ပါ။ Password ပြန်လာတောင်းရင် ပေးလိုက်ပါ။

ဒီနည်းနဲ့ App ဖန်တီးပြီးသွားတဲ့အခါ **Use cases** ကိုသွားပြီး **Customize** လုပ်လိုက်ပါ။



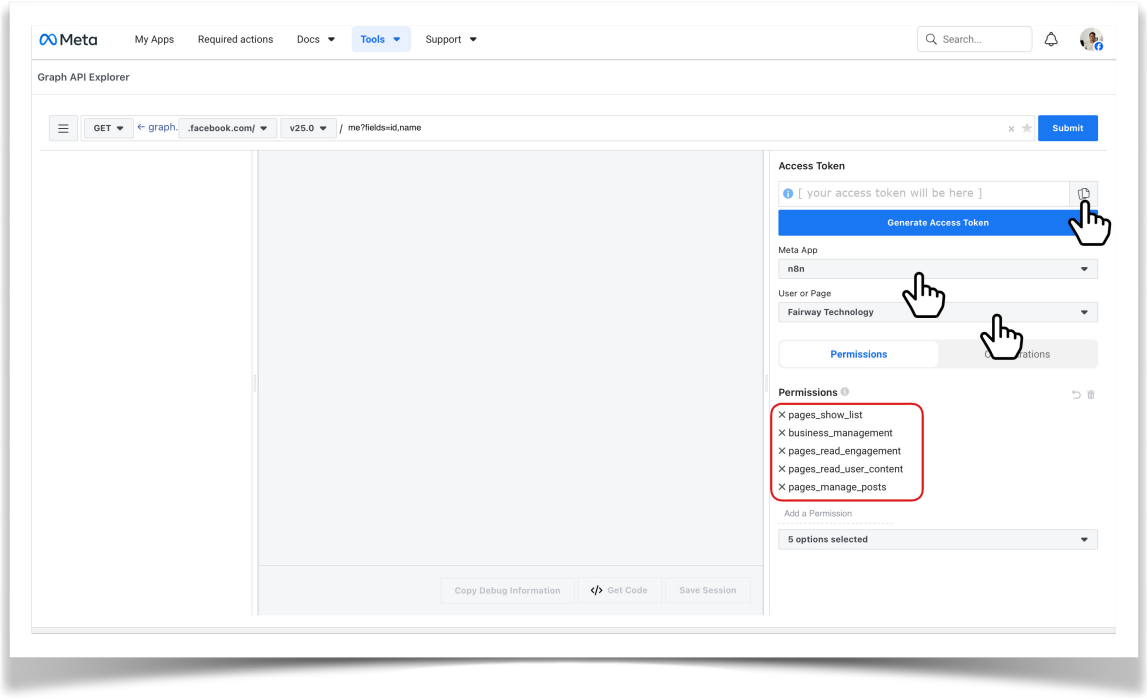
ပုံမှာဝိုင်းပြထားတာတွေ တစ်ခုမှ မကျန်ခဲ့အောင် အကုန် **+ Add** လုပ်ပေးလိုက်ပါ။



ဒီတော့မှ ဒီ App ကနေတစ်ဆင့် Page ရဲ့ Content တွေကို စီမံလို့ရသွားမှာပါ။

ပြီးတဲ့အခါ **Access Token** ရယူဖို့အတွက် **Meta Graph API Explorer** ကို ဒီလင့်ကနေသွားလိုက်ပါ။

https://developers.facebook.com/tools/explorer/



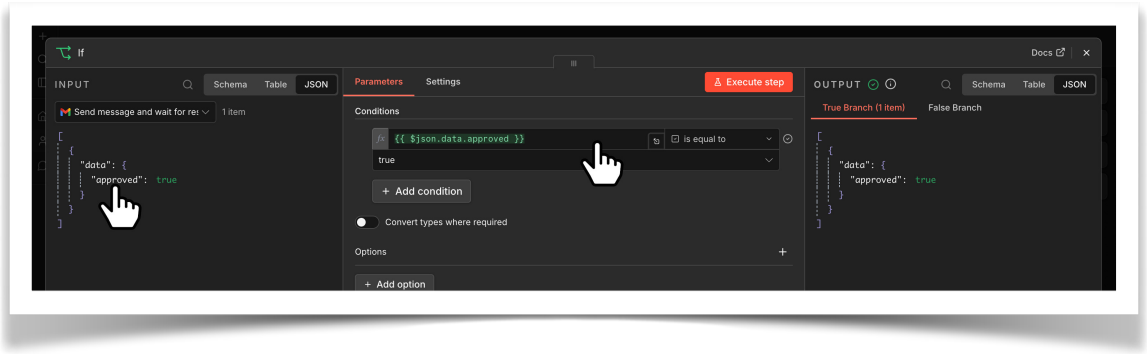
ပုံမှာဝိုင်းပြထားသလို Permissions တွေအကုန်စုံအောင် အရင်ရွေးလိုက်ပါ။ ပြီးရင် ကိုယ့် App နဲ့ Page ကို ရွေးပြီး **Generate Access Token** ကို နှိပ်လိုက်ပါ။ ပြီးတဲ့အခါ **Copy** ခလုတ်ကိုနှိပ်ပြီး ကူးယူထားလိုက်ပါ။

နောက်တစ်ဆင့်အနေနဲ့ ကိုယ့် Page ရဲ့ ID လိုပါသေးတယ်။ ကိုယ့် Facebook Page ကိုသွားလိုက်ပါ။ Cover ပုံအောက်က Page အမည်ကိုနှိပ်လိုက်ပါ။ ပေါ်လာတဲ့ထဲက

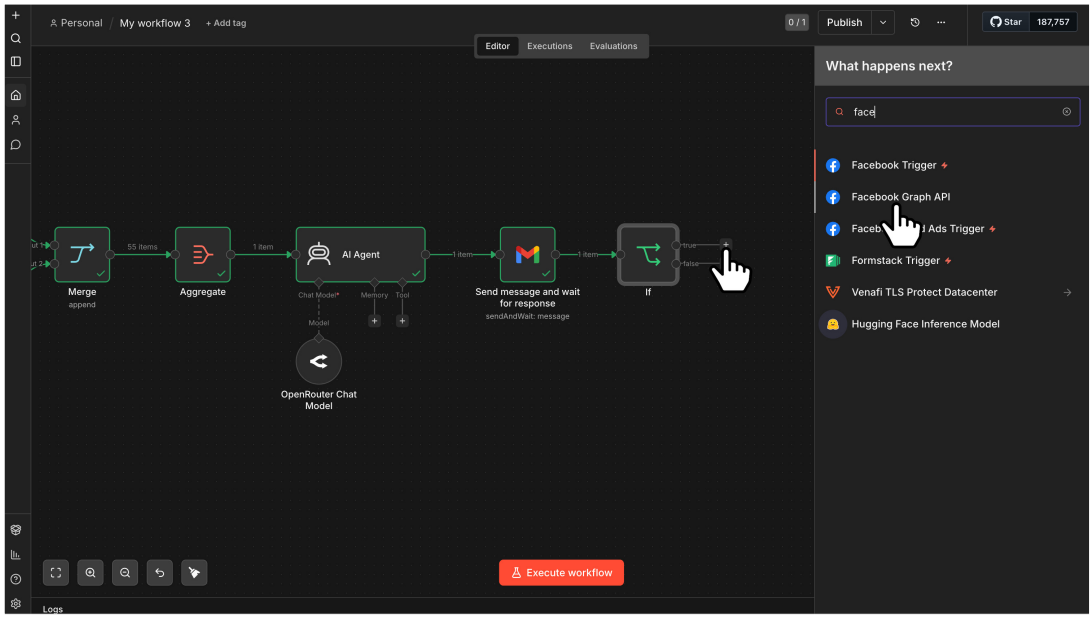
Transparency and privacy policy ကို နှိပ်လိုက်ပါ။ ဒါဆိုရင် တခြားအချက်အလက်တွေနဲ့အတူ **Page ID** ကိုလည်း တွေ့မြင်ရပါလိမ့်မယ်။ ကူးယူထားလိုက်ပါ။

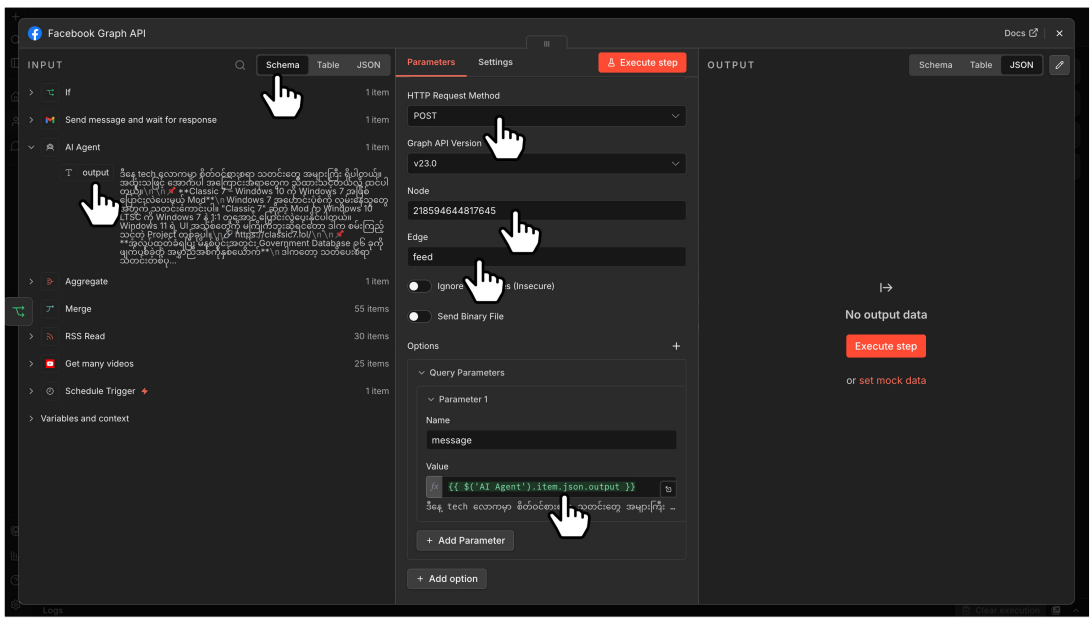
ဒီလောက်ဆိုရင် Facebook Page ကို n8n ကနေ Content တွေတင်တာ လုပ်လို့ရသွားပါပြီ။ လုပ်လက်စ Workflow ကို ပြန်သွားကြပါမယ်။

If Node တစ်ခုထည့်လိုက်ပါ။ စောစောက ထည့်ထားတဲ့ Email Approval က ပြန်လာတာ **true** ဆိုမှပဲ အလုပ်လုပ်ချင်တဲ့အတွက်ပါ။



ပြီးရင် **If Node** ရဲ့ True မှာ အခုလို **Facebook Graph API Node** ကို ထည့်ပေးလိုက်ပါ။ Credential ပေးဖို့အတွက် စောစောက ရထားတဲ့ Access Token ကို ပေးလိုက်ပါ။

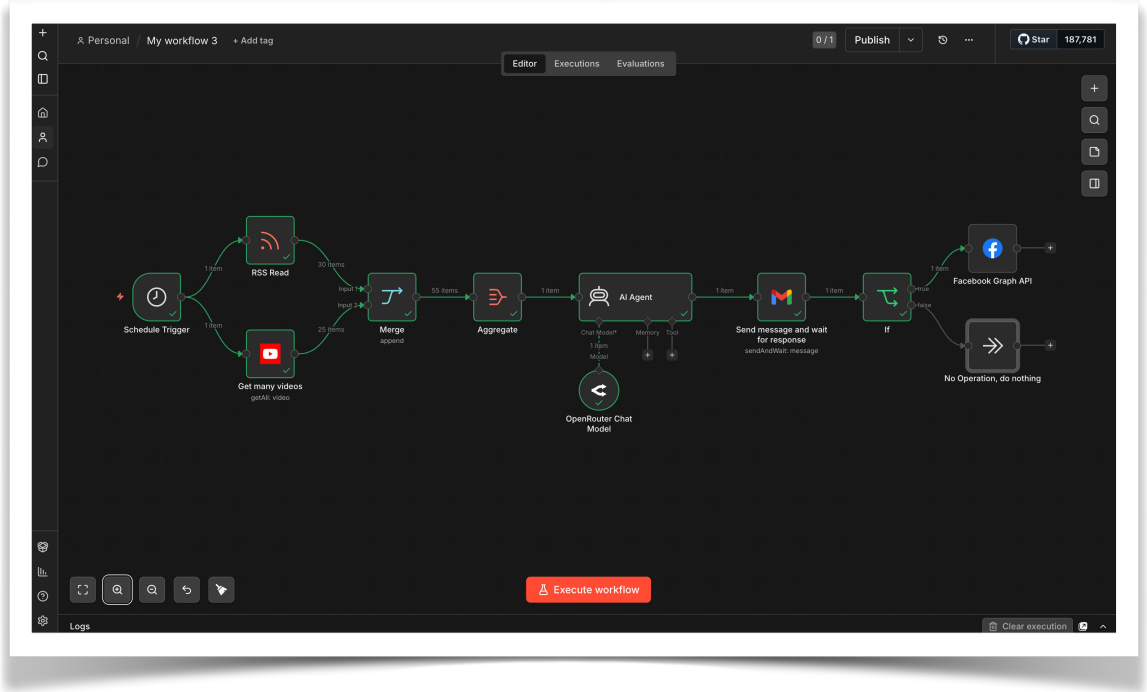

Personal / My workflow 3 + Add tag
Editor Executions Evaluations
0 / 1 Publish
Star 187,757
What happens next?
face
Facebook Trigger
Facebook Graph API
Ads Trigger
Formstack Trigger
Venafi TLS Protect Datacenter
Hugging Face Inference Model
Merge
append
55 items
Aggregate
1 item
AI Agent
Chat Model*
Memory
Tool
Model
OpenRouter Chat Model
1 item
Send message and wait for response
sendAndWait: message
1 item
If
true
false
Execute workflow
Logs


Facebook Graph API
Docs
INPUT
Schema Table JSON
If 1 item
Send message and wait for response 1 item
AI Agent 1 item
output
Aggregate 1 item
Merge 55 items
RSS Read 30 items
Get many videos 25 items
Schedule Trigger 1 item
Variables and context
Parameters Settings
Execute step
HTTP Request Method
POST
Graph API Version
v23.0
Node
218594644817645
Edge
feed
Ignore s (Insecure)
Send Binary File
Options
Query Parameters
Parameter 1
Name
message
Value
{{ $('AI Agent').item.json.output }}
+ Add Parameter
+ Add option
OUTPUT
Schema Table JSON
No output data
Execute step
or set mock data
Logs

HTTP Request Method ကို **POST** လို့ရွေးပါ။ Node နေရာမှာ ကိုယ့် **Page ID** ကိုထည့်ပါ။ နမူနာပုံက ID ကိုထည့်ရင် အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ Edge မှာ **feed** လို့ထည့်ပေးပါ။ ဒါမှ တင်လိုက်တဲ့ပို့စ်က Timeline Feed ပေါ်ရောက်သွားမှာပါ။

+ Add Option နဲ့ Query Parameter ထည့်ပေးပါ။ Name ကို **message** လို့ပေးပြီး Value နေရာမှာ AI Agent ကရတဲ့ Output ကိုဆွဲထည့်လိုက်ပါ။ ဒါပါပဲ။ ရသွားပါပြီ။

နောက်ဆုံးဖိုင်နယ် Workflow က အခုလိုပုံစံ ဖြစ်ပါလိမ့်မယ်။



ဒါဟာ AI Agent Node ကို အသုံးပြုပြီး နေ့စဉ် နောက်ဆုံးရသတင်းနဲ့ အချက်အလက်တွေပေါ်အခြေခံထားတဲ့ Content တစ်ခုကို အလိုအလျောက် Generate လုပ်ရုံသာမက၊

Facebook မှာလည်း အလိုအလျောက် တင်ပေးသွားနိုင်တဲ့ Workflow တစ်ခုကို အောင်မြင်စွာ ရရှိသွားတာပဲ ဖြစ်ပါတယ်။

နမူနာ Workflow တွေကို ကိုယ်တိုင်ကြိုးစားပြီး လုပ်ကြည့်ပါ။ လိုအပ်ရင်တော့ ဒီလင့်က နေလည်း Download ရယူနိုင်ကြောင်း ထပ်မံအသိပေးလိုက်ပါတယ်။

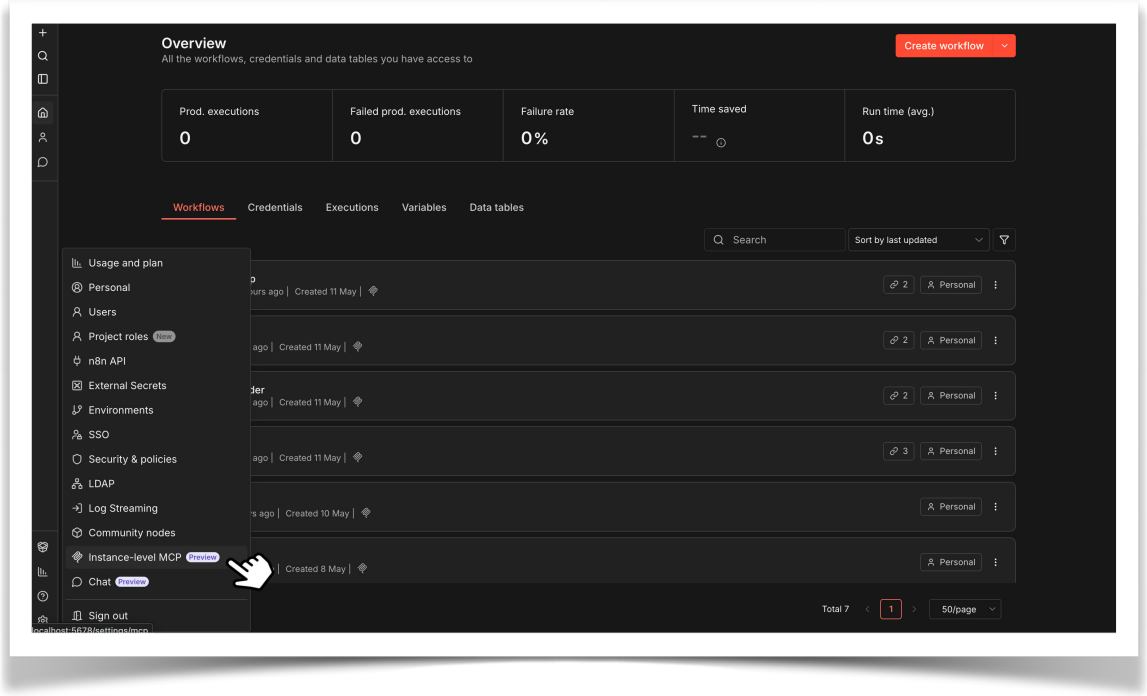
https://github.com/eimg/n8n-workflows/archive/refs/heads/main.zip

အခန်း (၁၀) - AI နှင့် Workflow များဖန်တီးခြင်း

အခုရှေ့မှာလေ့လာခဲ့တဲ့ Workflow တွေကို ကိုယ်တိုင်လုပ်ရင်လည်းရသလို AI ကို လုပ်ခိုင်းရင်လည်းရပါတယ်။ AI နည်းပညာတွေရဲ့ ထုံးစံအတိုင်း၊ ကိုယ်တိုင် လုပ်တတ်အောင် အရင်လေ့လာရပါတယ်။ လုပ်တတ်သွားပြီဆိုရင်တော့ အမြဲတမ်း ကိုယ်တိုင်လုပ်နေစရာ မလိုတော့ပါဘူး။

အခု n8n Workflow တွေဆိုရင်လည်း AI ကို Draft လုပ်ခိုင်းပြီး ကိုယ်က လိုအပ်တာလေးတွေ လိုက်ဖြည့် လက်စသတ်လို့ ရပါတယ်။ ဒါဆိုရင် အလုပ်တွေ တော်တော်မြန်သွားမှာ ဖြစ်သလို၊ တချို့ ခက်ခဲလို့ ကိုယ်တိုင်လုပ်ရခက်နေတာလေးတွေမှာလည်း အကူအညီ ရသွားမှာပါ။

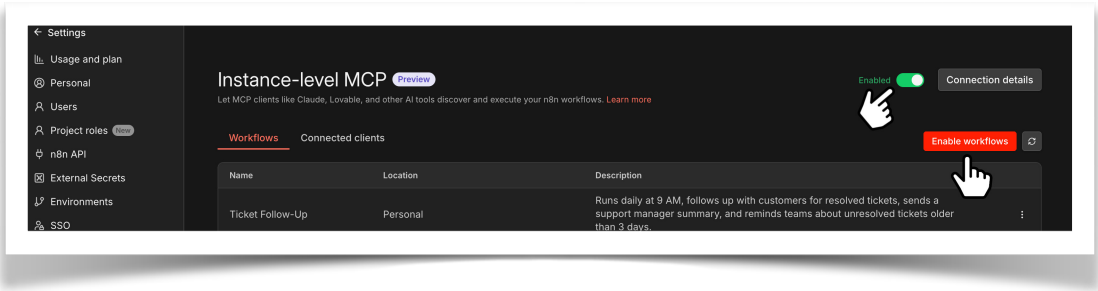
တော်တော် အသုံးဝင်ပါတယ်။ ဒီလိုပါ။



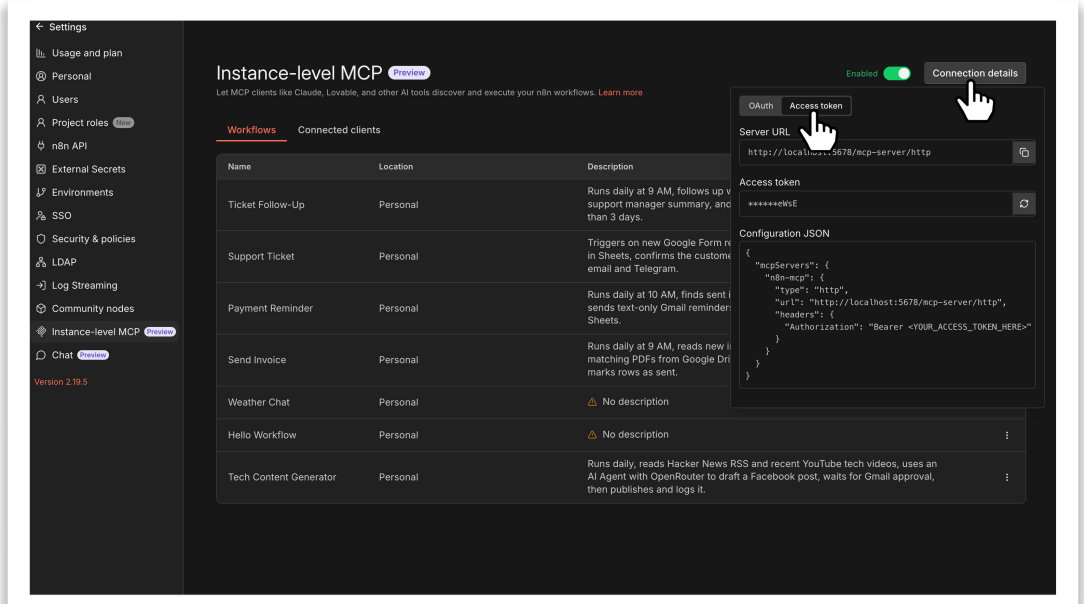
ဘယ်ဘက်အောက်နားလေးက **Settings** ခလုတ်ကိုနှိပ်ပြီး **Instant-level MCP** ကို ရွေးလိုက်ပါ။ MCP ဆိုတာ (Model Context Protocol) ခေါ်နည်းပညာဖြစ်ပြီး n8n ရဲ့ လုပ်ဆောင်ချက်တွေကို AI Agent တွေက လာချိတ်သုံးလို့ရအောင် ဖွင့်ပေးတဲ့ နည်းပညာတစ်မျိုးဖြစ်ပါတယ်။

MCP နဲ့ n8n ဘက်က ဖွင့်ပေးလိုက်ပြီးရင် မိမိနှစ်သက်ရာ AI Agent နဲ့ ဒီ n8n ကို လာချိတ်လို့ရသွားတာပါ။ Instant Level လို့သုံးလိုက်တာက လက်ရှိ Run ထားတဲ့ n8n တစ်ခုလုံးနဲ့ သက်ဆိုင်တယ်၊ စက်ထဲက တခြား n8n တွေ၊ Online အကောင့်တွေနဲ့ မဆိုင်ဘူးဆိုတဲ့ သဘောပါပဲ။

နောက်တစ်ဆင့်မှာ **Enable** ခလုတ်ကို ဖွင့်ပေးလိုက်ပါ။ ပြီးရင် **Enable workflows** ခလုတ်ကိုနှိပ်ပြီး လက်ရှိ ရှိနေတဲ့ Workflow တွေထဲက MCP ကနေ ဝင်ပြင်ခွင့်ပေးချင်တဲ့ Workflow တွေကို ရွေးပေးလို့ရပါတယ်။



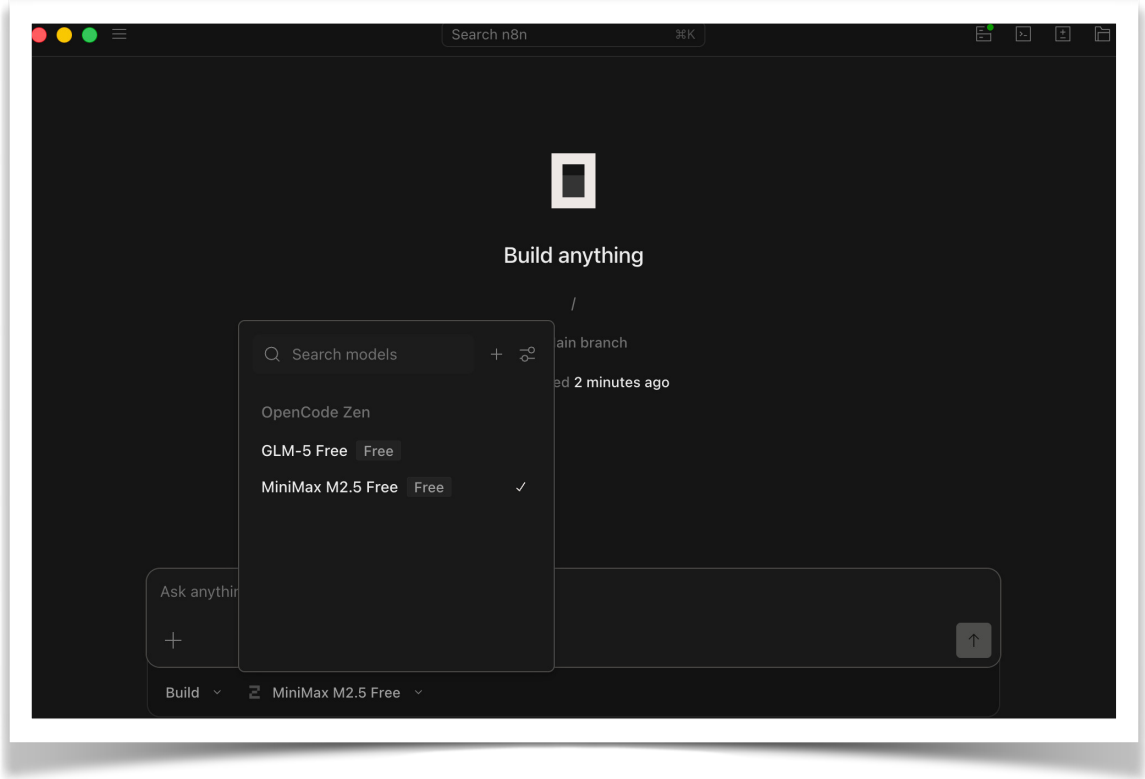
အဲ့ဒီလို ရွေးပေးထားမှသာ AI Agent တွေက လက်ရှိရှိနေတဲ့ Workflow တွေကို လာပြင်ခွင့် ရပါတယ်။ မဟုတ်ရင် ရမှာမဟုတ်ပါဘူး။ ဒါကြောင့် တချို့ Sensitive ဖြစ်တဲ့ အရေးကြီးတဲ့ Workflow တွေအတွက် ဖွင့်မပေးဘဲ ပိတ်ထားချင်ရင်လည်း ရပါတယ်။



နောက်တစ်ဆင့်အနေနဲ့ **Connection details** ကို နှိပ်ပါ။ **Access token** ကိုရွေးပြီး ပေါ်လာတဲ့ Information (၃) ခုလုံးကို ကူးယူထားလိုက်ပါ။ ဒါဆိုရင် n8n ဘက်က Ready ဖြစ်သွားပါပြီ။

စမ်းသပ်ကြည့်ဖို့အတွက် AI Agent အနေနဲ့ Claude Code, Codex, Cursor, OpenCode စတဲ့ နည်းပညာအမျိုးမျိုးရှိကြတဲ့ထဲက နှစ်သက်ရာကို အသုံးပြုနိုင်ပါတယ်။ ဒီနေရာမှာတော့ **OpenCode** နဲ့ နမူနာပြပါမယ်။ ဒီလင့်မှာ Download ရယူနိုင်ပါတယ်။

https://opencode.ai/download



OpenCode မှာ အခမဲ့ရတဲ့ Model တွေ ပါကြပါတယ်။ နမူနာမှာ **GLM-5 Free** နဲ့ **MiniMax M2.5 Free** နှစ်ခုပါဝင်လာရာမှာ GLM-5 Free က အလုပ်မလုပ်ပါဘူး။ အခမဲ့ရတဲ့ Model တွေက ဒီလိုပါပဲ ရတဲ့အခါလည်းရှိမယ်၊ မရတဲ့အခါလည်း ရှိတတ်ပါတယ်။

MiniMax M2.5 Free ကတော့ စမ်းကြည့်တဲ့အချိန်မှာ အဆင်ပြေလို့ ရွေးချယ် အသုံးပြုထားပါတယ်။ ပထမဆုံးအနေနဲ့ ဒီ Prompt လေး ပေးလိုက်ပါ။

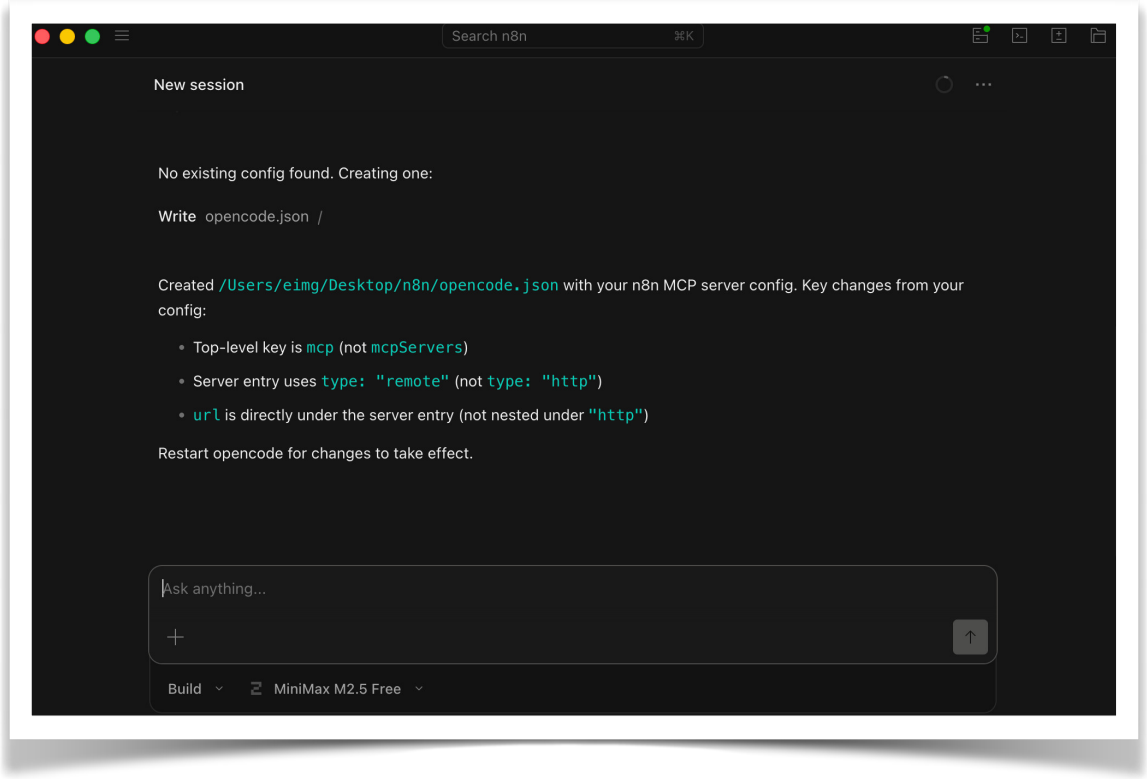
```
Help me setup MCP for local n8n in OpenCode, following is available
config, make sure it's up to OpenCode's config standard.

{
  "mcpServers": {
   …
  }
}
```

ငါ့ကို n8n MCP ထည့်ပေးပါ လို့ ခိုင်းလိုက်တာပါ။ Local n8n ဖြစ်ကြောင်း၊ OpenCode အတွက် ထည့်ရမှာဖြစ်ကြောင်း ပြည့်စုံအောင် ပြောထားပါတယ်။ စောစောက ကူးယူထားတဲ့ n8n MCP Config ကိုလည်း သူ့ကိုပေးလိုက်ပါတယ်။

တကယ်တော့ AI Agent မှာ MCP ထည့်တယ်ဆိုတာ ကိုယ့်ဘာသာ Setting တွေပြင်ပြီး ထည့်ရတာပါ။ အဲ့ဒါကို ကိုယ့်ဘာသာ ထည့်မနေတော့ဘဲ Agent ကိုပဲ ပြန်ခိုင်းလိုက်တာ ဖြစ်ပါတယ်။ AI က ကြည့်လုပ်သွားပါလိမ့်မယ်။

Setting ဖိုင်တွေ ပြင်ရမှာမို့လို့ ကိုယ့်ဆီက Permission လာတောင်းရင်သာ **Allow** လုပ်ပေးလိုက်ပါ။ စာရေးသူဆီမှာတော့ အားလုံးအဆင်ပြေပြေနဲ့ ရရှိသွားပါတယ်။



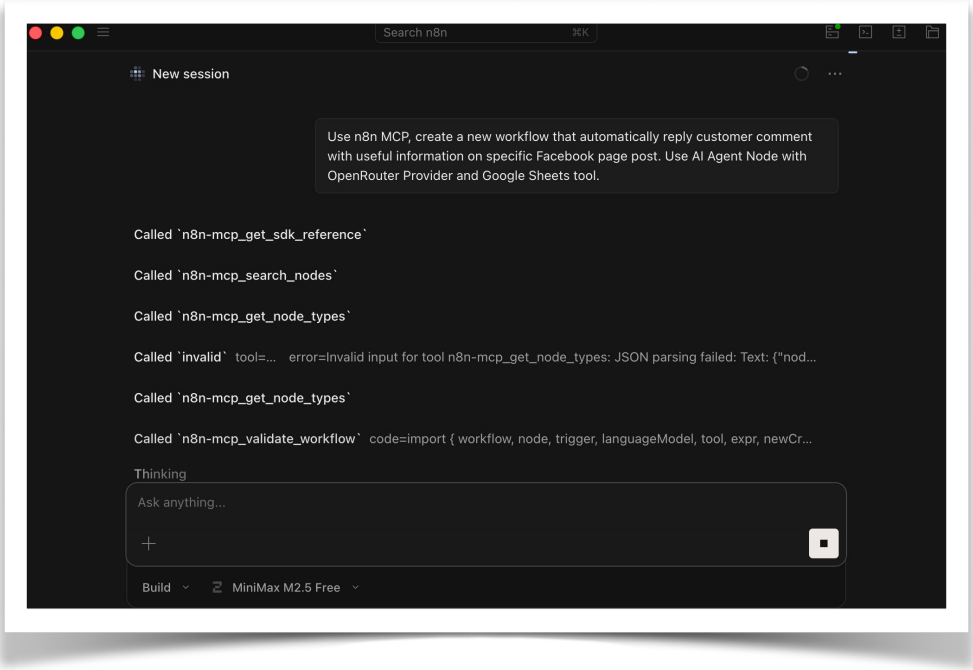
ပြီးရင် OpenCode ကို ပိတ်ပြီး ပြန်ဖွင့်လိုက်ပါ။ အခုမှ ထည့်လိုက်တဲ့ MCP က ဖွင့်လက်စ OpenCode မှာ အသက်မဝင်တာမျိုး ဖြစ်နေမှာစိုးလို့ပါ။ ပိတ်ပြီး ပြန်ဖွင့်လိုက်ရင်တော့ အသက်ဝင်ဖို့ သေချာသွားပါတယ်။

Workflow တွေစလုပ်ခိုင်းလို့ရပါပြီ။ ဒါကြောင့် အခုလို Prompt ပေးလိုက်ပါတယ်။

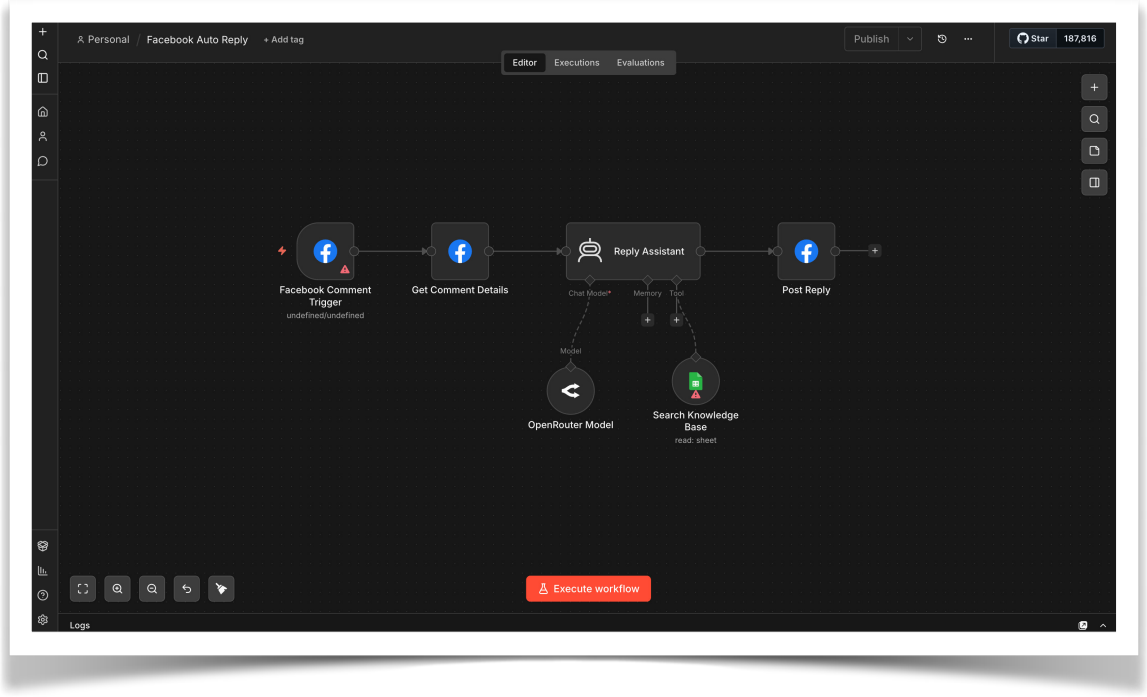
> *Use n8n MCP, create a new workflow that automatically reply customer comment with useful information on specific Facebook page post. Use AI Agent Node with OpenRouter Provider and Google Sheets tool.*

အင်္ဂလိပ်ဘာသာနဲ့ပဲ ပေးထားပါတယ်။ သေချာဂရုစိုက်ကြည့်လိုက်ပါ။ n8n MCP ကို သုံးပြီး အလုပ်လုပ်ဖို့ ပြောထားပါတယ်။ မပြောလည်း သူ့ဘာသာ အသုံးပြုနိုင်ပါတယ်။ ပို သေချာအောင် ပြောလိုက်တာပါ။

Facebook Post မှာ Comment တွေ အလိုအလျောက် Reply ပြန်ပေးတဲ့ Workflow လေး လုပ်ခိုင်းလိုက်တာပါ။ **AI Agent Node** ကို သုံးရမယ်။ **Google Sheets** tool တွဲပေးရမယ် စသည်ဖြင့် အသေးစိတ် ပြောထားပါတယ်။ ဒါမျိုးတွေကြောင့် AI ကိုလုပ်ခိုင်းမယ့် အလုပ်ကို ကိုယ်တိုင်လုပ်တတ်ရင် ပိုခိုင်းလို့ ကောင်းတာပါ။ ထုံးစံပါပဲ။



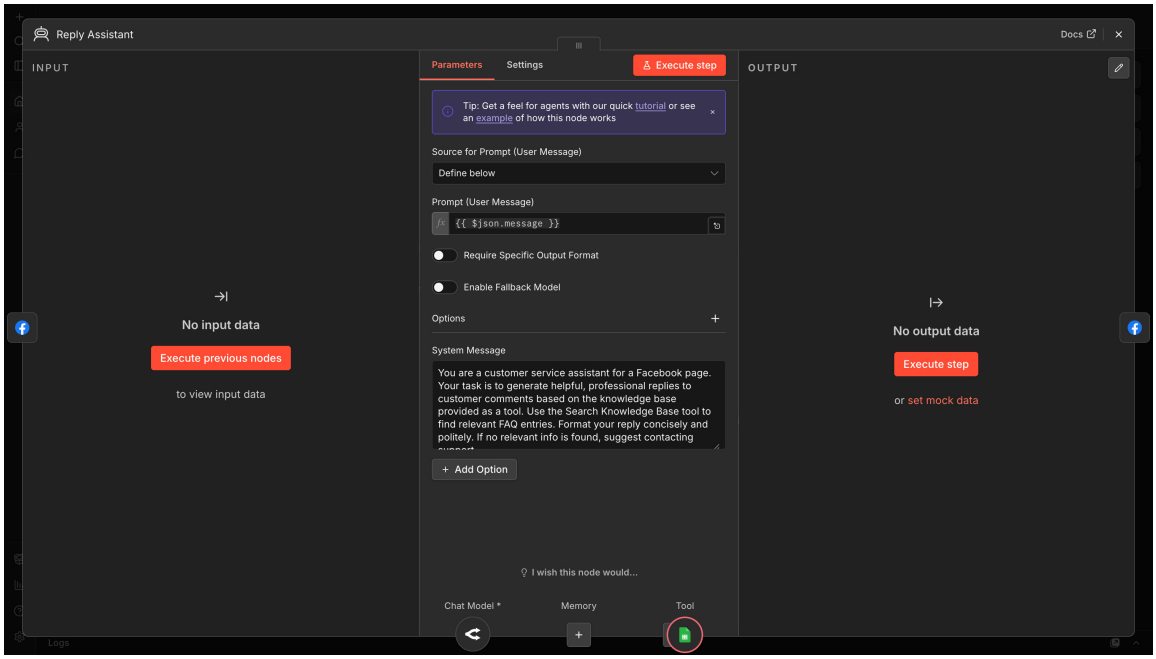
ပုံမှာတွေ့မြင်နေရသလို n8n MCP ကို လှမ်းခေါ်ပြီး အလုပ်တွေ လုပ်သွားတာကို တွေ့ရပါလိမ့်မယ်။ လုပ်ရင်းနဲ့ အမှားတွေရှိခဲ့ရင်လည်း၊ သူ့ဘာသာ တစ်ခါတည်း ပြင်သွားတာကို တွေ့ရနိုင်ပါတယ်။ စာရေးသူဆီမှာ အားလုံးအဆင်ပြေပြေ တစ်ချက်တည်းနဲ့ ရရှိသွားတဲ့ အတွက် n8n ဘက်ကို သွားကြည့်လိုက်တဲ့အခါ အခုလို Workflow တစ်ခု အသင့်ရရှိနေတာကို တွေ့မြင်ရပါတယ်။



ဒီလို တစ်ချက်တည်းနဲ့ အဆင်မပြေခဲ့ရင်လည်း စိတ်မပျက်ဘဲ၊ ပြန်ကြိုးစားကြည့်ပါ။ အခမဲ့ရတဲ့ Model ကိုအသုံးပြုနေတာမို့လို့ နည်းနည်းတော့ သည်းခံရပါတယ်။ GPT-5.5 တို့လို ရှေ့ဆောင် Model တွေကို သုံးနိုင်ရင်တော့ ရလဒ်ကောင်းတွေ ပိုရမှာပါ။ ရလာတဲ့ Workflow မှာ **Facebook Comment Trigger** ကို အသုံးပြုထားပါတယ်။ ဒါကြောင့် Facebook မှာ Comment ဝင်လာတာနဲ့ ဒီ Workflow က အသက်ဝင်သွားမှာပါ။ ကိုယ့်

ဘက်က Credential နဲ့ အချက်အလက်တွေ ထည့်ပေးဖို့တော့ လိုပါတယ်။ ကိုယ်ပြောလိုက်တဲ့အတိုင်း **Google Sheets** နဲ့ချိတ်ထားတဲ့ **AI Agent Node** လည်းပါပါတယ်။ Google Sheets ထဲမှာ Customer ရဲ့မေးခွန်းတွေကို ဖြေနိုင်ဖို့အတွက် ရှိသင့်တဲ့ အချက်အလက်တွေ ရှိအောင်တော့ ကိုယ်က ကြိုတင်ပြင်ဆင်ထားရမှာပါ။

ဒါကြောင့် ထုံးစံအတိုင်း AI က အရမ်းအသုံးဝင်ပါတယ်။ ဒါပေမဲ့ ကိုယ်မပါရင်တော့ မပြီးပါဘူး။ AI Agent မှာ အခုလို ရှိသင့်တဲ့ **Prompt** Parameter နဲ့ **System Prompt** ကို လည်း သူက အသင့်ထည့်ပေးထားပါသေးတယ်။



ဒီနည်းနဲ့ ရှေ့အခန်းတွေမှာ ကိုယ်တိုင်လေ့လာထားတဲ့ ဗဟုသုတကို AI နဲ့ ပေါင်းစပ်ပြီး ပိုကျယ်ပြန့်ပြည့်စုံတဲ့ Workflow တွေကို တည်ဆောက်သွားနိုင်မှာပဲ ဖြစ်ပါတယ်။

အခန်း (၁၁) - ngrok နှင့် အသုံးပြုခြင်း

အခန်း (၉) မှာလုပ်ခဲ့တဲ့ Content တွေ အလိုအလျောက် Generate လုပ်ပြီး Facebook မှာ တင်ပေးနိုင်တဲ့ Workflow ကို မှတ်မိကြပါလိမ့်မယ်။ အဲ့ဒီ Workflow မှာ Email ပေးပို့ပြီး Approval ယူတဲ့အဆင့် လည်းထည့်ခဲ့ကြပါတယ်။ လက်တွေ့မှာ ဒီလိုအကြောင်းကြားအတည်ပြုချက်ယူမှုမျိုးအတွက် Email ထက် Telegram လို နည်းပညာက ပိုသင့်တော်နိုင်ပါတယ်။

ဒါပေမဲ့ Telegram နဲ့ နမူနာမပြဘဲ Email နဲ့ နမူနာပြခဲ့တာ အကြောင်းရှိပါတယ်။ တချို့နည်းပညာတွေက ကိုယ့်စက်ထဲမှာ Run ထားတဲ့ n8n နဲ့ အလုပ်မလုပ်ပါဘူး။ လုံခြုံရေးအရ localhost ဖြစ်နေတဲ့အတွက် လက်ခံအလုပ်မလုပ်တာ ဖြစ်နိုင်သလို၊ HTTPS ကို မသုံးဘဲ ရိုးရိုး HTTP ကို အသုံးပြုထားလို့ လက်ခံအလုပ်မလုပ်တာလည်း ဖြစ်နိုင်ပါတယ်။

ဒါကြောင့် တကယ်တမ်း လုပ်ငန်းသုံး သုံးတော့မယ်ဆိုရင် n8n ကို ကိုယ့်စက်ထဲမှာ Install မလုပ်ဘဲ၊ VPS ဆာဗာတစ်လုံးနဲ့ Install လုပ်ထားဖို့ လိုနိုင်ပါတယ်။ ကိုယ့်စက်ထဲမှာဆိုရင်၊ ဒါမှမဟုတ် အလုပ်ထဲက ရိုးရိုးကွန်ပျူတာ တစ်လုံးဆိုရင်၊ Schedule နဲ့ Run ရမယ့် Workflow တွေက စက်ပိတ်ထားချိန် ဖြစ်နေလို့ Run မရဘူးဆိုတဲ့ ပြဿနာမျိုးတွေ ရှိလာ

နိုင်ပါတယ်။ ကွန်ပျူတာမှာ အင်တာနက်လိုင်းကျနေလို့ Run မရဘူး ဆိုတာမျိုးတွေ ဖြစ်နိုင်ပါတယ်။ VPS ဆာဗာ တစ်လုံးနဲ့ ဖြစ်သွားမှ ဒီလို စက်အဖွင့်အပိတ်ကိစ္စတွေ၊ အင်တာနက်လိုင်းကိစ္စတွေကို လိုက်ကြည့်နေစရာမလိုဘဲ 24/7 အသင့်ဖြစ်နေမှာပါ။

ပြီးတော့ အဲ့ဒီ VPS ဆာဗာကို Domain Name တစ်ခုနဲ့လည်း ချိတ်ထားပေးဖို့ လိုနိုင်ပါတယ်။ Telegram အပါအဝင် တချို့ Third-Party Service တွေ အလုပ်လုပ်ဖို့အတွက် **Webhook** တွေ လိုကြပါတယ်။

Webhook ဆိုတာ သက်ဆိုင်ရာ Service က အလုပ်လုပ်ပြီးရင်၊ ကိုယ့်စက်ကို ပြန်လည်ဆက်သွယ်နိုင်တဲ့လင့် လို့ပဲ အလွယ်မှတ်နိုင်ပါတယ်။ ကိုယ့်စက်က အင်တာနက်ပေါ်က တကယ့်ဆာဗာမဟုတ်ဘဲ localhost ဖြစ်နေလို့ Webhook ကနေ ပြန်လည်ဆက်သွယ်လို့ မရတဲ့အတွက် အလုပ်မလုပ်နိုင်ဘူးဆိုတာ ဖြစ်နိုင်ပါတယ်။

ဒီလိုအကြောင်းအမျိုးမျိုးတွေကြောင့် အမှန်တကယ် အသုံးချတော့မယ်ဆိုရင် n8n ကို အနည်းဆုံး VPS ဆာဗာတစ်လုံးလောက်နဲ့ တပ်ဆင်အသုံးပြုဖို့ကို စဉ်းစားသင့်ပါတယ်။

ဒီနေရာမှာတော့ VPS အကြောင်း အသေးစိတ် ထည့်မပြောနိုင်တော့ပါဘူး။ n8n ကို VPS မှာ Install လုပ်နည်းက လွယ်ပါတယ်။ ကိုယ့်စက်ထဲမှာ Install လုပ်သလိုပဲ Docker နဲ့ ဖြစ်ဖြစ်၊ NPM နဲ့ဖြစ်ဖြစ် VPS မှာလည်း Install လုပ်လို့ရပါတယ်။

VPS ဆာဗာကို စီမံတဲ့နည်းတွေက ပြောဖို့ခက်သွားတာပါ။ Linux Server Administration ပိုင်းတွေ ပြောရတော့မှာမို့လို့ပါ။ ဒီအကြောင်းတွေကို **OpenClaw လိုတိုရှင်း** စာအုပ်မှာ

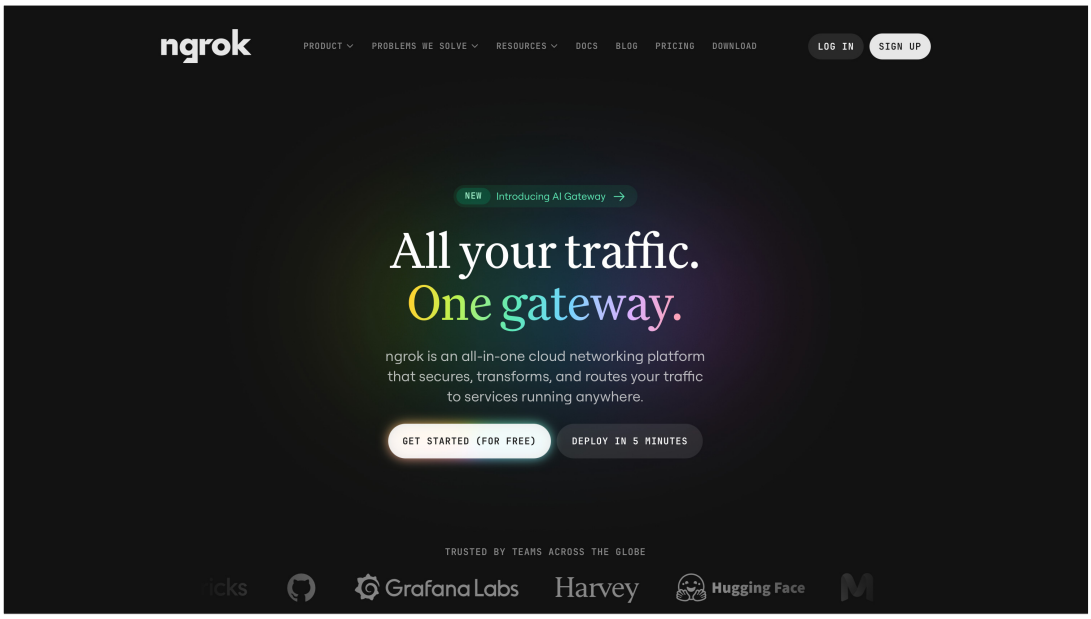
ဖော်ပြထားပြီး ဖြစ်ပါတယ်။ ဒါကြောင့် လိုအပ်ရင် **OpenClaw လိုတိုရှင်း** စာအုပ်မှာ ဆက်လက် လေ့လာကြည့်လိုက်ပါ။

http://eimaung.com/openclaw

ဒီနေရာမှာတော့ VPS တွေဘာတွေ မလိုသေးဘဲ ကိုယ့်စက်ထဲမှာပဲ Webhook အမှန်လိုတဲ့ နည်းပညာတွေကို ဘယ်လိုစမ်းလို့ရမလဲ ပြောပြပါမယ်။ **ngrok** လို့ခေါ်တဲ့ နည်းပညာတစ်ခုကို အသုံးပြုနိုင်ပါတယ်။ အောက်ပါလင့်ကို သွားလိုက်ပါ။

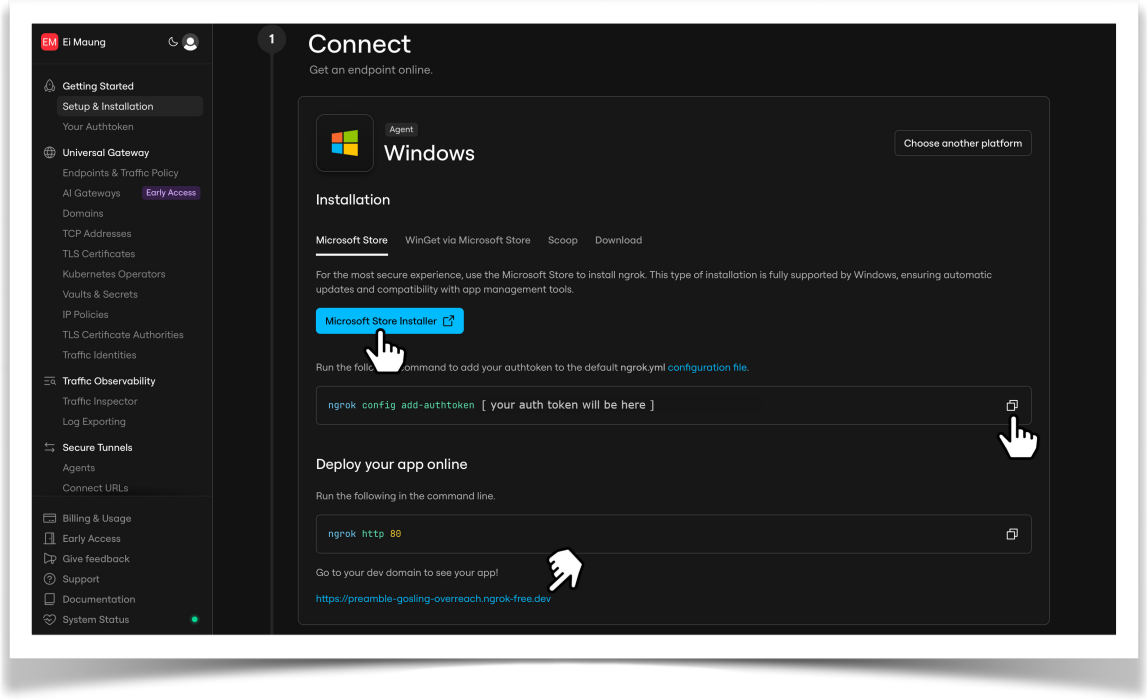
https://ngrok.com/

ပြည်တွင်းကနေ အသုံးပြုလို့ရဖို့ဆိုရင် VPN လိုကောင်းလိုနိုင်ပါတယ်။



Setting Up ngrok

အကောင့်သစ်ဆောက်လိုက်ရင် ရသလို၊ Google Account နဲ့လည်း Login ဝင်လိုက်လို့ရ ပါတယ်။ အသုံးပြုခ အခကြေးငွေ ပေးစရာမလိုပါဘူး။ အကောင့် Login ဝင်လိုက်ရင် အသုံးပြုနည်း လာပြပါလိမ့်မယ်။ စာဖတ်သူက Windows သုံးနေရင် အောက်ကပုံမှာ ပြ ထားသလို Instruction ကို တွေ့မြင်ရမှာပါ။ Mac တို့ Linux တို့ အသုံးပြုတာ ဆိုရင်လည်း သင့်တော်တဲ့ Instruction ကို ပြပေးပါလိမ့်မယ်။



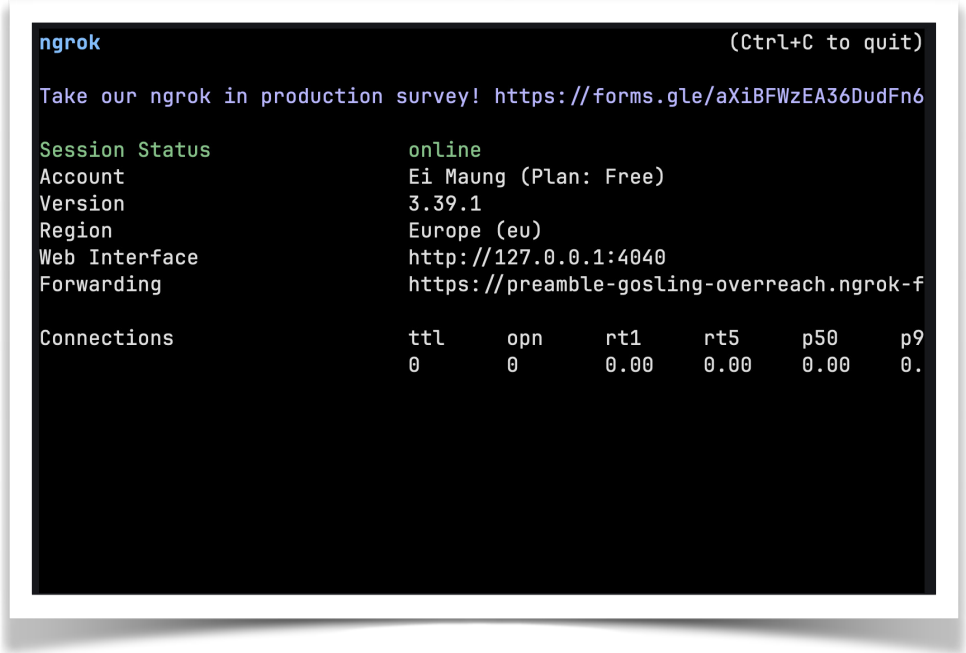
ပထမတစ်ဆင့်အနေနဲ့ ngrok ကို ကိုယ့်စက်ထဲမှာ Install လုပ်လိုက်ပါ။ Install လုပ်ပြီး သွားရင် နောက်တစ်ဆင့်အနေနဲ့ သူအသင့်ပြင်ဆင်ပေးထားတဲ့ `ngrok config` နဲ့ စတဲ့ Command ကို ကူးယူပြီး Terminal မှာ Run လိုက်ပါ။

ဟိုးအောက်နားလေးက **.dev** Domain Name လေးကို အထူးသတိပြုပါ။ အဲ့ဒါလေး ကူးယူထားလိုက်ပါ။ ngrok က ကိုယ့်အတွက် ဖန်တီးပေးထားတဲ့ ယာယီ Domain Name ဖြစ်ပါတယ်။ ဒီအထိရရင် ngrok ကို Setup လုပ်လို့ ပြီးသွားပါပြီ။

နောက်တစ်ဆင့်အနေနဲ့ Terminal မှာ အခုလို Run ပေးလိုက်ပါ။

```
ngrok http 5678
```

အခုလိုရလဒ်ကို တွေ့မြင်ရပါလိမ့်မယ်။



ဒါက ngrok ကို ကိုယ့်စက်ရဲ့ Port 5678 နဲ့ ချိတ်ပေးလိုက်တာပါ။ ngrok Domain Name ကို ဝင်လိုက်ရင် ကိုယ့်စက်ရဲ့ localhost:5678 ကို ရောက်သွားမှာ ဖြစ်ပါတယ်။ အဲ့ဒီလို ရောက်လာအောင် သူက Tunnel တစ်ခုအနေနဲ့ အလုပ်လုပ်ပေးသွားတာပါ။

Docker + ngrok

n8n ကို အရင်လို ဒီအတိုင်း Run လို့တော့ မရတော့ပါဘူး။ Run ပုံ Run နည်း နည်းနည်း ပြောင်းရပါမယ်။ Docker Command နဲ့ Run ချင်တာဆိုရင် နောက်ထပ် Terminal တစ်ခု ထပ်ဖွင့်လိုက်ပြီး Mac တို့ Linux တို့မှာဆိုရင် အခုလို Run ရပါမယ်။

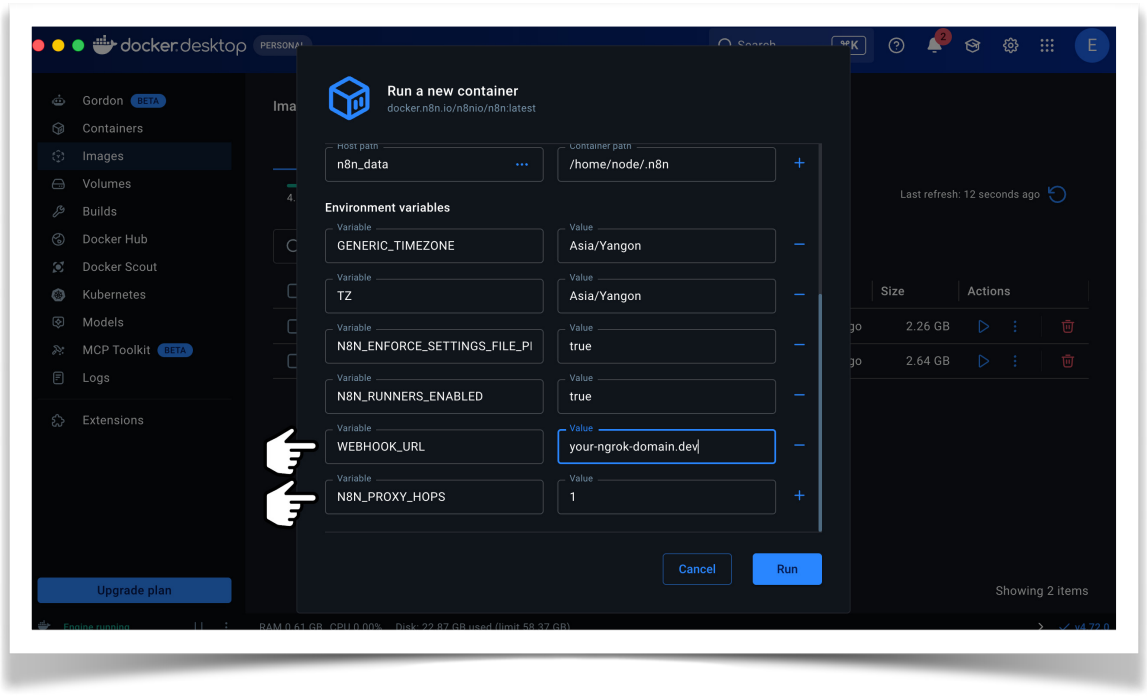
```
docker run -it --rm \
  --name n8n \
  -p 5678:5678 \
  -e GENERIC_TIMEZONE="Asia/Yangon" \
  -e TZ="Asia/Yangon" \
  -e N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true \
  -e N8N_RUNNERS_ENABLED=true \
  -e WEBHOOK_URL="your-ngrok-domain.dev" \
  -e N8N_PROXY_HOPS=1 \
  -v n8n_data:/home/node/.n8n \
  docker.n8n.io/n8nio/n8n
```

Windows မှာဆိုရင်တော့ အခုလို Run ရပါတယ်။

```
docker run -it --rm --name n8n -p 5678:5678 -e
GENERIC_TIMEZONE=Asia/Yangon -e TZ=Asia/Yangon -e
N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true -e
N8N_RUNNERS_ENABLED=true -e WEBHOOK_URL=your-ngrok-
domain.dev -e N8N_PROXY_HOPS=1 -v n8n_data:/home/
node/.n8n docker.n8n.io/n8nio/n8n
```

WEBHOOK_URL နဲ့ **N8N_PROXY_HOPS** ဆိုတဲ့နှစ်ခု ထပ်တိုးသွားတာပါ။ your-ngrok-domain.dev နေရာမှာ ngrok ဆီကရထားတဲ့ ကိုယ့် ယာယီ Domain Name အမှန်ကို ထည့်ပေးရမှာပါ။

Docker Desktop ကနေ Run တာဆိုရင်လည်း အခုလို ထည့်ပေးရမှာပါ။



မူလပေးရလေ့ရှိတဲ့ Variable တွေနဲ့အတူ **WEBHOOK_URL** နဲ့ **N8N_PROXY_HOPS** တို့ကို ထပ်တိုးပေးလိုက်ရတာပါ။

NPM + ngrok

NPM နဲ့ Install လုပ်ထားတဲ့ n8n ကို Run ဖို့အတွက်ဆိုရင်တော့ အခုလို ကြိုတင်ပြင်ဆင်ရပါတယ်။ Run လက်စ ngrok ကို ရပ်လို့မရဘဲ၊ နောက်ထပ် Terminal အသစ်တစ်ခုနဲ့ Run ရတယ်ဆိုတာ သတိပြုပါ။

```
npm i -g cross-env
```

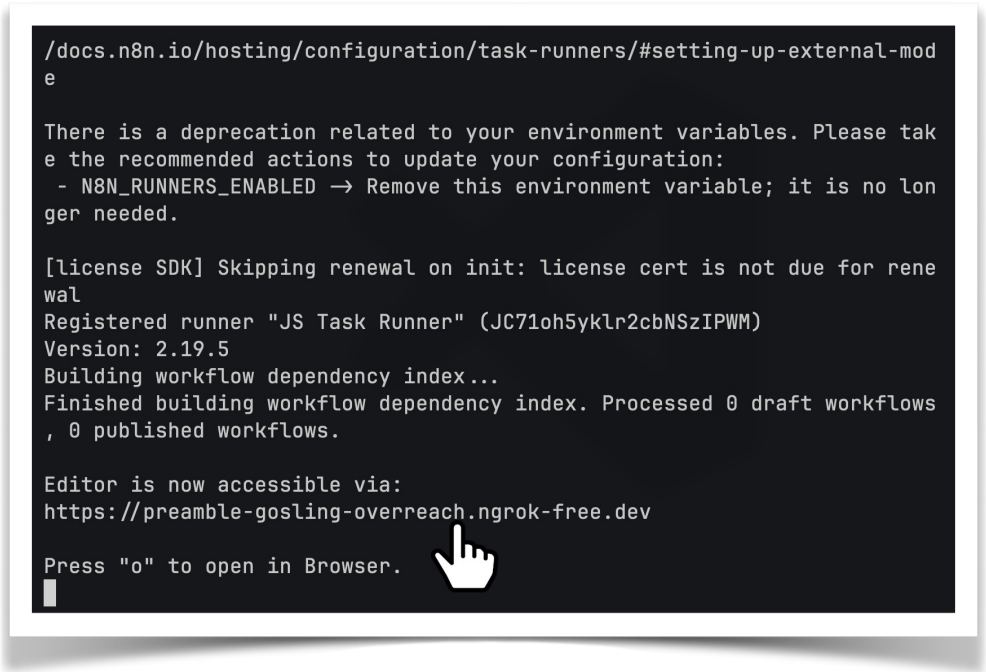
NPM နဲ့ Install လုပ်ထားတဲ့ n8n ကို Run ပုံ Run နည်းက Windows တို့ Mac တို့မှာ မတူကြပါဘူး။ အဲ့ဒါတွေကြောင့် ခေါင်းစားစရာမလိုအောင် cross-env လို့ခေါ်တဲ့ နည်းပညာကို Install လုပ်လိုက်တာပါ။ သူနဲ့ဆိုရင် Windows ပဲဖြစ်ဖြစ်၊ Mac ပဲဖြစ်ဖြစ်၊ Linux ပဲဖြစ်ဖြစ် Run နည်းတူသွားပါတယ်။ အခုလို Run ရပါတယ်။

```
cross-env WEBHOOK_URL=your-ngrok-domain.dev N8N_PROXY_HOPS=1 n8n
```

n8n ကို တိုက်ရိုက် မ Run ဘဲ လိုအပ်တဲ့ **WEBHOOK_URL** နဲ့ **N8N_PROXY_HOPS** ကို အရင်သတ်မှတ်ပြီးမှ Run လိုက်တာပါ။ ပိုပြည့်စုံချင်ရင် အခုလို Run ရမှာပါ။

```
cross-env GENERIC_TIMEZONE=Asia/Yangon TZ=Asia/Yangon
N8N_ENFORCE_SETTINGS_FILE_PERMISSIONS=true
N8N_RUNNERS_ENABLED=true WEBHOOK_URL=your-ngrok-domain.dev
N8N_PROXY_HOPS=1 n8n
```

ဒီတော့မှ ပါသင့်တဲ့ Timezone Setting တွေဘာတွေ စုံအောင် ပါသွားမှာပါ။ အခုလို ရလဒ်ကို ပြန်ရပါလိမ့်မယ်။



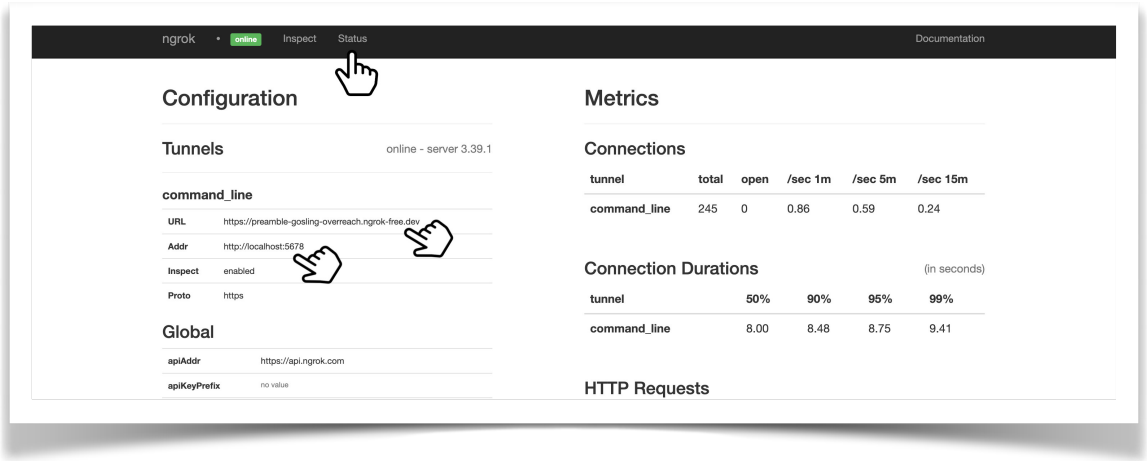
Run သွားတဲ့ n8n ကို **localhost:5678** ကနေ မသွားခိုင်းတော့ဘဲ ကိုယ်ချိတ်ပေးလိုက်တဲ့ Domain Name ကနေ သွားခိုင်းနေတာကို တွေ့ရပါလိမ့်မယ်။

နမူနာမှာ တွေ့မြင်နေရတာကတော့ ကျွန်တော်စာရေးသူရဲ့ ngrok URL ဖြစ်ပါတယ်။ စာဖတ်သူဆီမှာတော့ စာဖတ်သူ ထည့်ပေးလိုက်တဲ့ ကိုယ်ပိုင် URL ကို တွေ့မြင်ရမှာပါ။

Keyboard ကနေ အို "O" ကို နှိပ်ပြီး Browser မှာကြည့်ရင်ရသလို၊ ကိုယ့်ဘာသာ ကိုယ့်ရဲ့ ngrok Domain Name ကို Browser မှာ ကိုယ့်ဘာသာစမ်းကြည့်ရင်လည်း ရပါတယ်။ သူက ဝင်မှာသေချာလား Confirmation တောင်းတဲ့သဘော မေးလာခဲ့ရင် Confirm လုပ်ပေးလိုက်ဖို့တော့ လိုအပ်ပါတယ်။

ဒီလိပ်စာကိုလည်း သွားကြည့်လိုက်ပါ။

http://localhost:4040



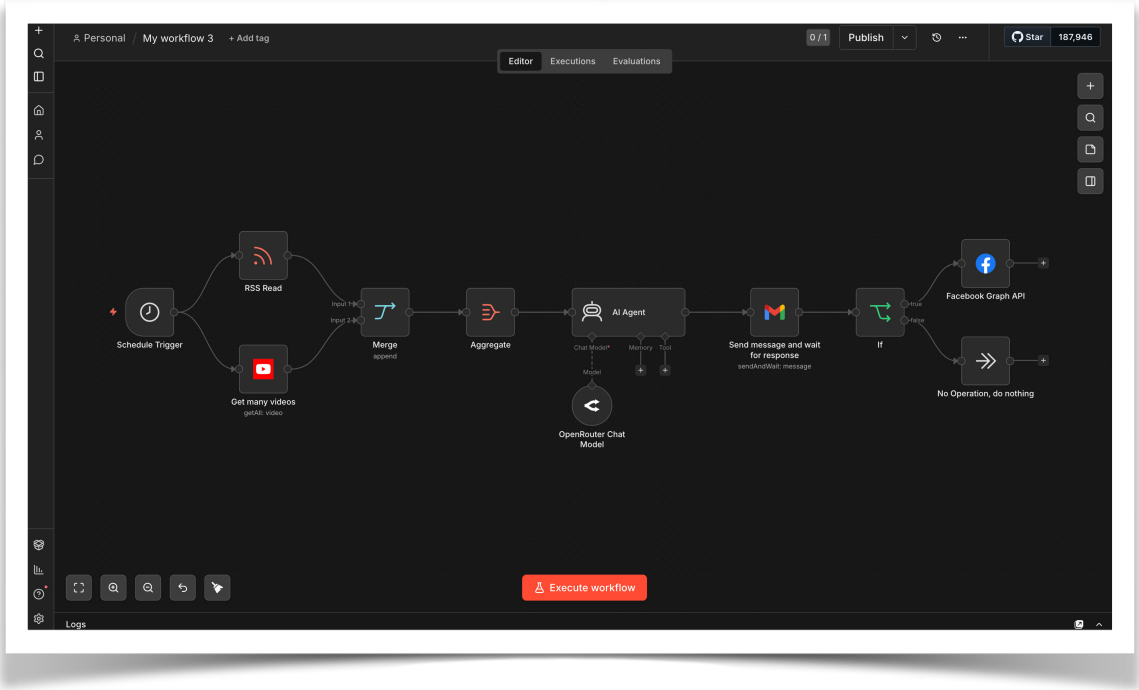
ngrok Domain Name နဲ့ localhost:5678 တို့ကို ချိတ်ဆက်ထားရှိမှု အခြေအနေတွေ့ လာပြနေတာပါ။

ဒီနည်းနဲ့ ကိုယ့်စက်ထဲမှာပဲ n8n ကို HTTPS တွေ ကိုယ်ပိုင် ယာယီ Domain Name တွေနဲ့ Run လိုက်တာ ဖြစ်သွားပါတယ်။ ဒါကြောင့် Webhook လုပ်ဆောင်ချက်တွေလည်း အလုပ်လုပ်သွားပါလိမ့်မယ်။

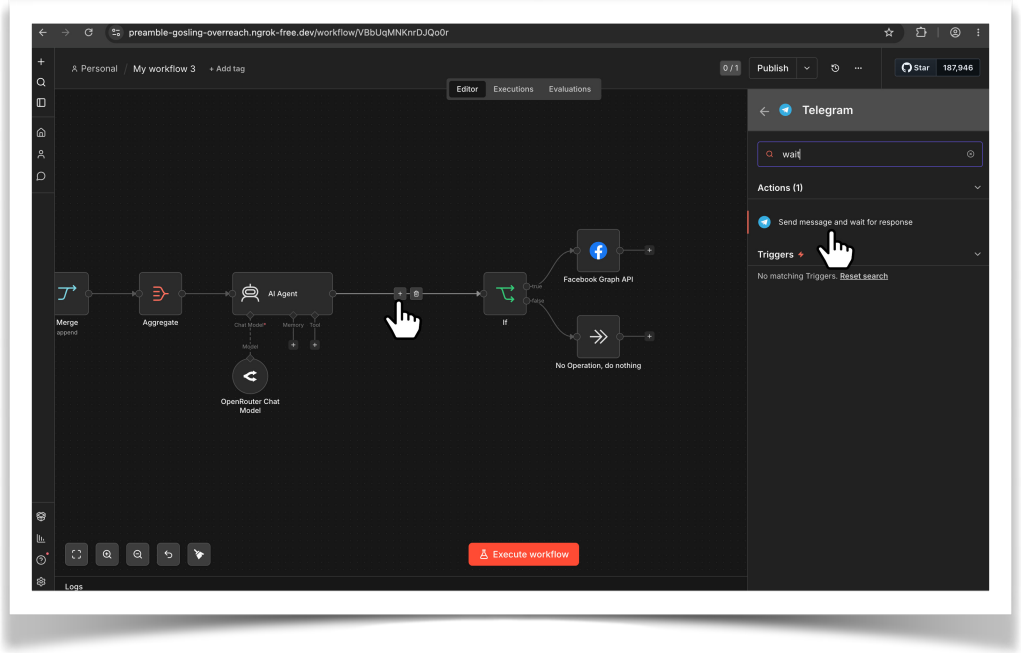
ဒါကို လက်တွေ့စမ်းသပ်ကြည့်တဲ့အနေနဲ့ အခန်း (၉) မှာ လုပ်ခဲ့တဲ့ Workflow ကို ပြင်ဆင်ကြည့်ကြပါမယ်။

Telegram Approval Workflow

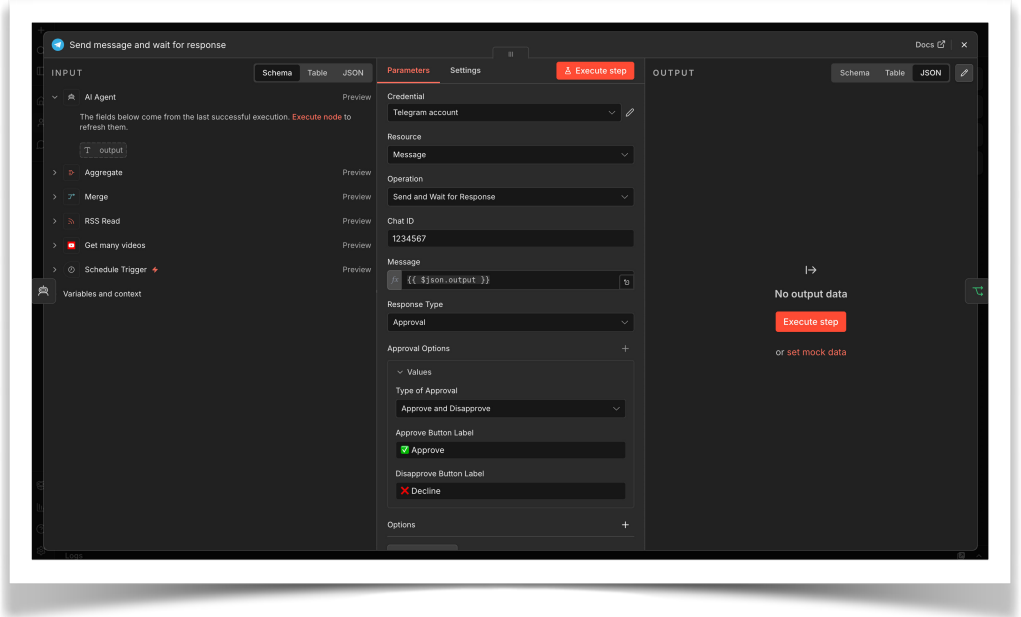
အခန်း (၉) မှာ လုပ်ခဲ့တဲ့ Workflow ရဲ့ လက်ရှိအခြေအနေက ဒီလိုပါ။



အဓိကပြင်ချင်တာက **Gmail Node** နဲ့ Approval တစ်ဆင့်ခံထားတာကို **Telegram** နဲ့ အစားထိုးချင်တာပါ။ ဒါကြောင့် အခုလို ပြင်လိုက်ပါ။



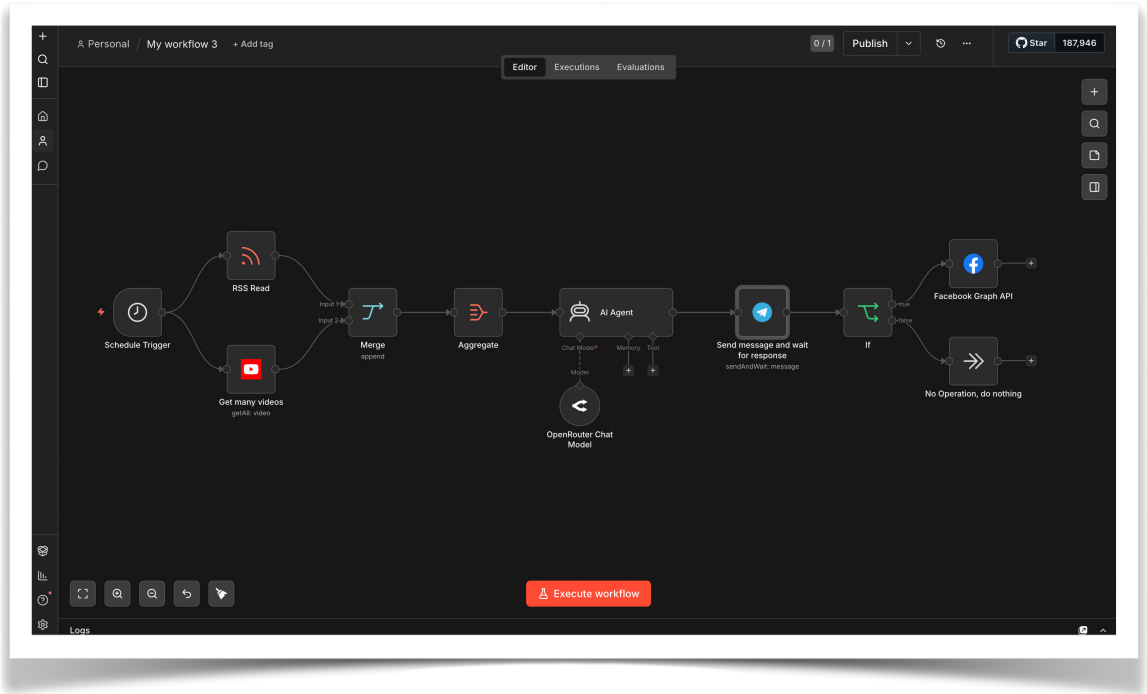
Gmail Node ကို ဖျက်လိုက်ပြီး **Telegram** ရဲ့ **Send message and wait for response** ကို ထည့်လိုက်တာပါ။



Credential ကတော့ ပြီးခဲ့တဲ့အခန်းတွေမှာ ထည့်ခဲ့ဖူးရင် ထပ်ထည့်စရာ မလိုတော့ပါဘူး။ **Chat ID** မှာသာ ကိုယ့် Chat ID အမှန်ကို ထည့်ပေးလိုက်ပါ။ ပြီးရင် **+ Add Option** ခလုတ်ကိုနှိပ်ပြီး **Approve and Disapprove** ကိုလည်း ရွေးထားပေးလိုက်ပါ။

ဒါဆိုရင် ရသွားပြီ။ မူလက localhost ဖြစ်နေလို့ Gmail နဲ့ပဲ စမ်းခဲ့တဲ့ Approval Workflow ဟာအခုဆိုရင် Telegram နဲ့လည်း အလားတူ အလုပ်လုပ်သွားပြီပဲ ဖြစ်ပါတယ်။

နောက်ဆုံးရလဒ်က အခုလိုပုံစံ ဖြစ်ပါလိမ့်မယ်။



လက်တွေ့စမ်းသပ်ကြည့်လိုက်ပါ။ အီးမေးလ်ကို ပို့သလိုပဲ AI Agent ရေးပေးထားတဲ့ Content ကို Approve လုပ်ပေးဖို့ ကိုယ့်ဆီကို Telegram ကနေ ပို့သွားတာ တွေ့မြင်ရမှာပဲ ဖြစ်ပါတယ်။

ကိုယ့်က **Approve** လုပ်ပေးလိုက်မှသာ **If Node** ကို ဖြတ်ပြီး Facebook မှာ ပို့စ်အနေနဲ့ တင်ပေးသွားမှာပဲ ဖြစ်ပါတယ်။

ကွန်ဂရက်ကျူးလေးရှင်းပါ။ ဒီအထိရပြီဆိုရင် ကျွန်တော်တို့ ဒီစာအုပ်မှာ လေ့လာချင်တဲ့ အကြောင်းအရာတွေအားလုံး ပြည့်စုံသွားပြီပဲ ဖြစ်ပါတယ်။

ဒီစာအုပ်မှာ အသုံးပြုခဲ့တဲ့ Workflow နမူနာတွေကို Download ရယူနိုင်တဲ့လင့်ကို ထပ်မံ ဖော်ပြပေးလိုက်ပါတယ်။

https://github.com/eimg/n8n-workflows/archive/refs/heads/main.zip

နိဂုံးချုပ်

Workflow Automation ဆိုတာ နေရာစုံမှာ အသုံးဝင်နိုင်တဲ့ နည်းပညာဖြစ်ကြောင်းကို နမူနာတွေနဲ့အတူ တွေ့မြင်ခဲ့ကြပါပြီ။ လုပ်ငန်းအရွယ်အစားမရွေး Sales, Marketing, Admin, HR, IT စသည်ဖြင့် လုပ်ငန်းဌာနအသီးသီးမှာ အသုံးချလို့ ရနိုင်ပါတယ်။ ဒီနေရာကနေ အကြံပြုချင်တာလေးတချို့ ရှိပါတယ်။

ရှိသမျှအလုပ်အားလုံးကို Automate လုပ်ထားဖို့ဆိုတာ မလွယ်ပါဘူး။ တချို့ ဘယ်လိုမှ Automate လုပ်လို့မရနိုင်တဲ့ အပိုင်းတွေဟာ ရှိနေမှာပါ။ အားလုံးကို Automate လုပ်ပေးနိုင်တဲ့ ဧရာမ Workflow ကြီးတွေထက်၊ လုပ်ငန်းအစိတ်အပိုင်းလိုက် သီးခြားစီ Automate လုပ်ပေးနိုင်တဲ့ Workflow လေးတွေက ပိုလက်တွေ့ကျပြီး ပိုအသုံးဝင်တာမျိုး ဖြစ်နိုင်ပါတယ်။ အစပိုင်းမှာ အကုန်လုံးကို Automate လုပ်ပစ်ဖို့ ရည်ရွယ်တာမျိုးထက်၊ Automate လုပ်ဖို့ အသင့်တော်ဆုံး အပိုင်းလေးတချို့ကို ရွေးထုတ်ပြီးအစပြုသင့်ပါတယ်။

တစ်ခါတစ်ရံမှာ ကိုယ့်လုပ်ငန်းသဘောတရားက ထူးခြားလွန်းလို့ အသင့်ရနိုင်တဲ့ Node တွေ၊ Service တွေနဲ့ လိုချင်တဲ့အတိုင်း စိတ်တိုင်းကျမရနိုင်တာမျိုးဟာလည်း ဖြစ်နိုင်ပါတယ်။ အဲ့ဒီလို အခြေအနေမျိုးမှာ Code တွေရေးဖို့ မဖြစ်မနေ လိုလာတာ၊ Custom API

တွေဖန်တီးဖို့ လိုလာတာမျိုးလည်း ဖြစ်နိုင်ပါတယ်။ ဆန္ဒရှိရင် **Vibe Coding** လို သဘောတရားမျိုးတွေ လေ့လာပြီး၊ AI အကူအညီနဲ့ လိုအပ်တာတွေ ရေးသားဖန်တီးလို့ အတိုင်းအတာတစ်ခုထိ ရနိုင်ပါတယ်။ ဒါပေမဲ့ ကျွမ်းကျင်သူတွေရဲ့ အကူအညီကို ယူဖို့လို လာတာမျိုးလည်း ရှိနိုင်ပါတယ်။

ပြီးတော့ Run တဲ့ Workflow တွေများလာမယ်၊ လုပ်ရတဲ့အလုပ်တွေ များလာမယ်ဆိုရင် n8n ကို Default အတိုင်း Setup လုပ်ဖို့ မသင့်တော်တော့ဘဲ Database နည်းပညာတွေ အသုံးပြု Setup လုပ်ရတာမျိုးတွေ ရှိလာနိုင်ပါတယ်။ VPS ဆာဗာတွေနဲ့ Setup လုပ် မယ်ဆိုရင်လည်း Backup တို့ Maintenance တို့လိုကိစ္စတွေ၊ လုံခြုံရေးပိုင်း ကိစ္စတွေ စဉ်းစားစရာ ရှိလာပါလိမ့်မယ်။

ဒီစာအုပ်ရဲ့ ရည်ရွယ်ချက်က၊ နည်းပညာပိုင်း အဓိကလေ့လာနေတာ မဟုတ်တဲ့ လုပ်ငန်း ရှင်တွေကိုယ်တိုင် ကိုယ့်အလုပ်အတွက် Workflow တွေကို ကိုယ်တိုင်တည်ဆောက်နိုင်စေ ဖို့ထိ ရည်ရွယ်ပါတယ်။ အခု ဒီစာအုပ်အဆုံးထိရောက်လာပြီဖြစ်လို့ အတိုင်းအတာတစ်ခု ထိ လုပ်လည်းလုပ်နိုင်သွားပြီလို့လည်း ယူဆပါတယ်။

ဒါပေမဲ့ နည်းပညာပိုင်းမှာ Workflow တွေသာမက၊ အဲ့ဒီ Workflow တွေကို Run ပေးနိုင် တဲ့ Infrastructure ပိုင်းအနေနဲ့ ထပ်မံစဉ်းစားစရာလေးတွေ ကျန်နိုင်သေးတယ် ဆိုတာကို ချင့်ချိန်နိုင်ဖို့ ဖြည့်စွက်သတိပေးလိုက်တာပါ။

OpenClaw vs n8n

အလုပ်တွေကို အလိုအလျောက် လုပ်ပေးနိုင်တာချင်းအတူတူ၊ နောက်ထပ်လူပြောများတဲ့ နည်းပညာတစ်ခုဖြစ်တဲ့ OpenClaw လို AI Agent နည်းပညာမျိုး နဲ့ အခုလေ့လာလက်စ ဖြစ်တဲ့ n8n လိုနည်းပညာမျိုးတွေက ဘာတွေကွာတာလဲ။ ဘယ်ဟာကပိုကောင်းလဲ။ ဒါ လေးကိုလည်း ထည့်ပြောပြချင်ပါတယ်။ အကျဉ်းချုပ်လေး ဒီလိုမှတ်ပါ။

n8n Workflow တွေက ကြိုတင်သတ်မှတ်ထားတဲ့အတိုင်း တိတိကျကျ အလုပ်လုပ်ကြပါ တယ်။ OpenClaw လို AI Agent မျိုးကတော့ ဘာလုပ်ရမလဲဆိုတာ AI က ဆုံးဖြတ်ပြီး လုပ်သွားပါတယ်။ ဒါဟာ **အဓိက** ကွာခြားချက်ပါပဲ။

A → B → C ဆိုတဲ့ Workflow တစ်ခု ရှိတယ်ဆိုရင် သူ့ရဲ့အလုပ်လုပ်ပုံက ရှင်းပါတယ်။ A ပြီးရင် B ကို လုပ်မယ်။ B ပြီးရင် C ကို ဆက်လုပ်ပါမယ်။ ဘယ်နှစ်ကြိမ်ပဲ Run သည်ဖြစ် စေ၊ ဒီအတိုင်း အတိအကျ လုပ်သွားမှာပါ။

ဒီနေရာမှာသာ OpenClaw လို AI Agent ဆိုရင် ကိုယ်က "C ကို လိုချင်တယ်" လို့ ပြော လိုက်တဲ့အခါ၊ AI က C ကို ရဖို့အတွက် A ကို အရင်လုပ်ရမယ်၊ ပြီးရင် B ကို ဆက်လုပ်ရ မယ်ဆိုတာ သူ့ဘာသာ အဖြေရှာဆုံးဖြတ်ပြီး လုပ်ပေးသွားမှာပါ။ တစ်ခါတစ်လေ မှားပြီး B ကို အရင်လုပ်မိရင်လည်း သူ့ဘာသာ ပြင်ပြီး မှန်အောင် ပြန်လုပ်ပေးသွားမှာပါ။

လုပ်ငန်းပိုင်း Operation တွေဟာ တိကျတဲ့ အဆင့်တွေ Process တွေနဲ့ သွားကြတာများ လို့ လုပ်ငန်းပိုင်း Operation တွေအတွက်ဆိုရင် တိတိကျကျ ကြိုတင်သတ်မှတ်ထားတဲ့ Workflow တွေက ပိုသင့်တော်နိုင်ပါတယ်။

ဒီ Workflow တွေရဲ့ အားနည်းချက်ကတော့၊ ဒေတာအဝင်အထွက် ဖွဲ့စည်းပုံတွေ ပြောင်းလို့မရခြင်းနဲ့၊ ဒေတာဖွဲ့စည်းပုံ ပြောင်းသွားရင် Workflow ကို အလိုအလျောက် ပြောင်းလဲအလုပ်လုပ် မပေးနိုင်ခြင်းဖြစ်ပါတယ်။ တိကျတဲ့ ဒေတာအဝင်အထွက်နဲ့ သွားနေတဲ့ ကိစ္စတွေမှာတော့ ဒါဟာ ပြဿနာမဟုတ်ပါဘူး။

AI Agent တွေကတော့ ပြောင်းပြန် ဖြစ်သွားပါတယ်။ လုပ်နေရင်းနဲ့ ဘာဆက်လုပ်သင့်လဲ ဆုံးဖြတ်သွားတာဖြစ်လို့ ဒေတာဖွဲ့စည်းပုံတွေ ပြောင်းသွားတဲ့အခါ အလုပ်လုပ်ပုံတွေကိုလည်း အလိုက်သင့်ပြောင်းလဲ လုပ်ဆောင်ပေးနိုင်ပါလိမ့်မယ်။ အလုပ်အမျိုးအစား တူပေမဲ့ ဒေတာအဝင်အထွက် ပြောင်းလဲနေဖို့ရှိတဲ့ အလုပ်တွေနဲ့ ပိုသင့်တော်ပါတယ်။

တစ်ခါတစ်လေ ဖောက်ပြီး အမှားလုပ်သွားနိုင်တာတွေ၊ Hallucination လို ပြဿနာမျိုးတွေ၊ Prompt Injection လို အန္တရာယ်တွေလည်း ရှိနေပါသေးတယ်။

ဒါကြောင့် ဘယ်ဟာကပိုကောင်းတယ်ဆိုတာ မရှိဘဲ သူ့အားသာချက်နဲ့သူ၊ သူ့အားနည်းချက်နဲ့သူသာလျှင် ဖြစ်ပါတယ်။ တိတိကျကျလုပ်ရတဲ့ လုပ်ငန်းပိုင်းတွေမှာ Workflow တွေနဲ့ သင့်တော်ပြီး၊ ဖန်တီးမှုပါဖို့လိုတဲ့ အလုပ်တွေမှာ AI Agent တွေနဲ့ ပိုသင့်တော်တယ်လို့ အကျဉ်းချုပ် မှတ်နိုင်ပါတယ်။

n8n ရဲ့ အားသာချက်ကတော့ Workflow တွေထဲမှာလည်း AI Agent တွေကို ထည့်သွင်းအသုံးပြုနိုင်ခြင်း ဖြစ်ပါတယ်။ ဒီတော့ တိတိကျကျ အလုပ်လုပ်တဲ့ Workflow တွေနဲ့ အတူ Smart ဖြစ်ပြီး ဖန်တီးမှုအားကောင်းတဲ့ လုပ်ဆောင်ချက်တွေကိုလည်း ပူးတွဲ ရရှိသွားပါတယ်။

အပြောင်းအလဲတွေ များလှတဲ့ AI ခေတ်ဟာ အခုမှ စခါစပဲ ရှိပါသေးတယ်။ နည်းပညာတွေဟာလည်း အခြေတည်ခါစပဲရှိပါသေးတယ်။ အထိုင်မကျကြသေးပါဘူး။ ဒါပေမဲ့ အခြေကျအောင်ထိလည်း ထိုင်စောင့်နေစရာတော့ မလိုအပ်ပါဘူး။ နိုင်သလောက် လေ့လာ၊ လေ့လာမိသလောက် ပေါင်းစပ် အသုံးချလို့ ရပါတယ်။

အခု ဒီစာအုပ်မှာ လေ့လာခဲ့ကြတဲ့ n8n ကိုလည်း ထိရောက်အောင် အသုံးချကြပါ။ ကိုယ့် လိုအပ်ချက်နဲ့ ကိုက်ညီရင်၊ Vibe Coding တို့ OpenClaw တို့ကို နည်းပညာတွေကိုလည်း ပူးတွဲ လေ့လာကြပါလို့ တိုက်တွန်းရင်း နိဂုံးချုပ်လိုက်ပါတယ်။

အားလုံးပဲ အစစအရာရာ အဆင်ပြေကြပါစေ။

- **Vibe Coding လိုတိုရှင်း** - https://eimaung.com/vibe-coding/
- **OpenClaw လိုတိုရှင်း** - https://eimaung.com/openclaw/
- **n8n လိုတိုရှင်း** - https://eimaung.com/n8n/

အိမောင်

Fairway

မေလ (၁၇) ရက်၊ ၂၀၂၆ ခုနှစ်တွင် ရေးသားပြီးစီးသည်။