AWS Global Infrastructure

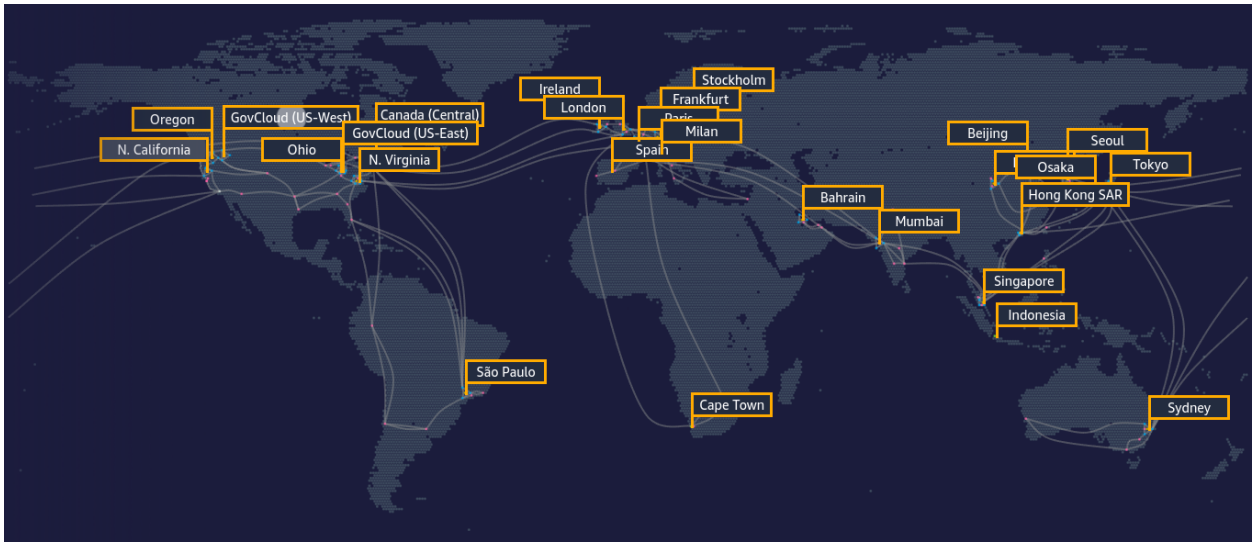
Global Infrastructure

ဒီနေ့မှာတော့ Cloud Service Provider တွေထဲကမှတစ်ခုဖြစ်တဲ့ Amazon Web Service က သူတို့ရဲ့ Global Infrastructure တွေကိုဘယ်လိုမျိုးတွေတည်ဆောက်ထားတယ်ဆိုတဲ့အကြောင်းကို ပြောပြပါရစေ။ အရင်ဦးဆုံးအနေနဲ့ Global Infrastructure ဆိုတဲ့အတိုင်း AWS ရဲ့ Infrastructure တွေကို ကမ္ဘာ့ geographic location တော်တော်များများမှာတည်ဆောက်ထားတာဖြစ်ပါတယ်။

1. Regions and Availability Zones

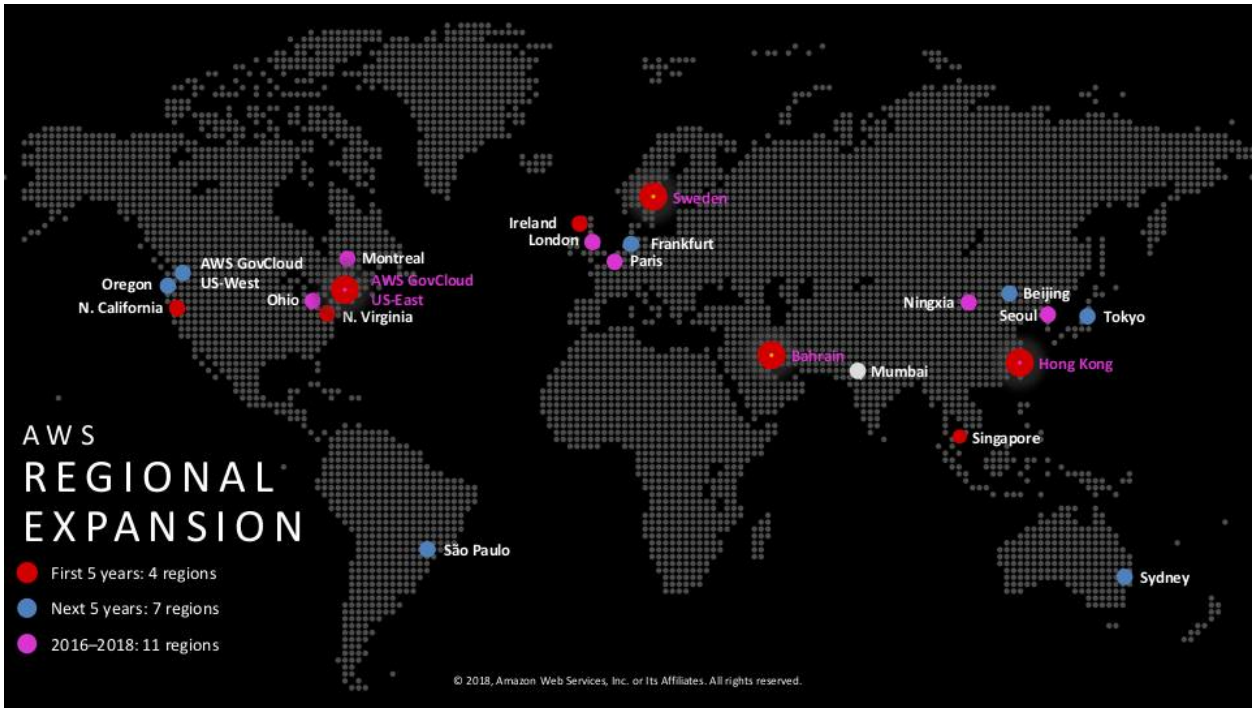
1.1.Regions

Geographic location (Region) တွေအနေနဲ့ အခု (၂၀) ရှိပြီးတော့ Availability Zone(AZs) အနေနဲ့ (၅၈) ခုရှိပါတယ်။ Region တစ်ခုခြင်းဆီမှာ ၂ ခုထက်ပိုတဲ့ Availability Zone တွေရှိကြပါတယ်။ Region တစ်ခုခြင်းဆီမှာ အနည်းဆုံး Data Center ၂ ခုရှိတဲ့သဘောပါ။ တစ်ချို့တစ်ချို့သော Region တွေကိုတော့ public access ပေးမလုပ်ထားပါဘူး။ Region တစ်ခုခြင်းဆီအနေနဲ့လည်း Isolate ဖြစ်ပါတယ်။ နောက်ပြီးတော့ Region တစ်ခုခြင်းဆီမှာအသုံးပြုနိုင်မဲ့ Services တွေအနေနဲ့လည်းမတူညီကြပါဘူး။ ဥပမာဆိုရင် AWS Global Infrastructure မှာအကြီးဆုံး Region ဖြစ်တဲ့ US-East (N. Virginia) မှာရနိုင်တဲ့ Core Services တွေကို တစ်ချို့တစ်ချို့သော Regions တွေမှာဆိုရင်မရရှိနိုင်သေးပါဘူး။ ဒီအချက်ကတော့ ကျွန်တော်တို့ရဲ့ ကိုယ်ပိုင် Cloud Infrastructure ကို စိတ်ကြိုက် architect ပြုလုပ်ရာမှာ အဓိကထည့်သွင်းစဉ်းစားဖို့လိုအပ်တဲ့တစ်ချက်ဖြစ်ပါတယ်။



Regions

AWS Regional Expansion

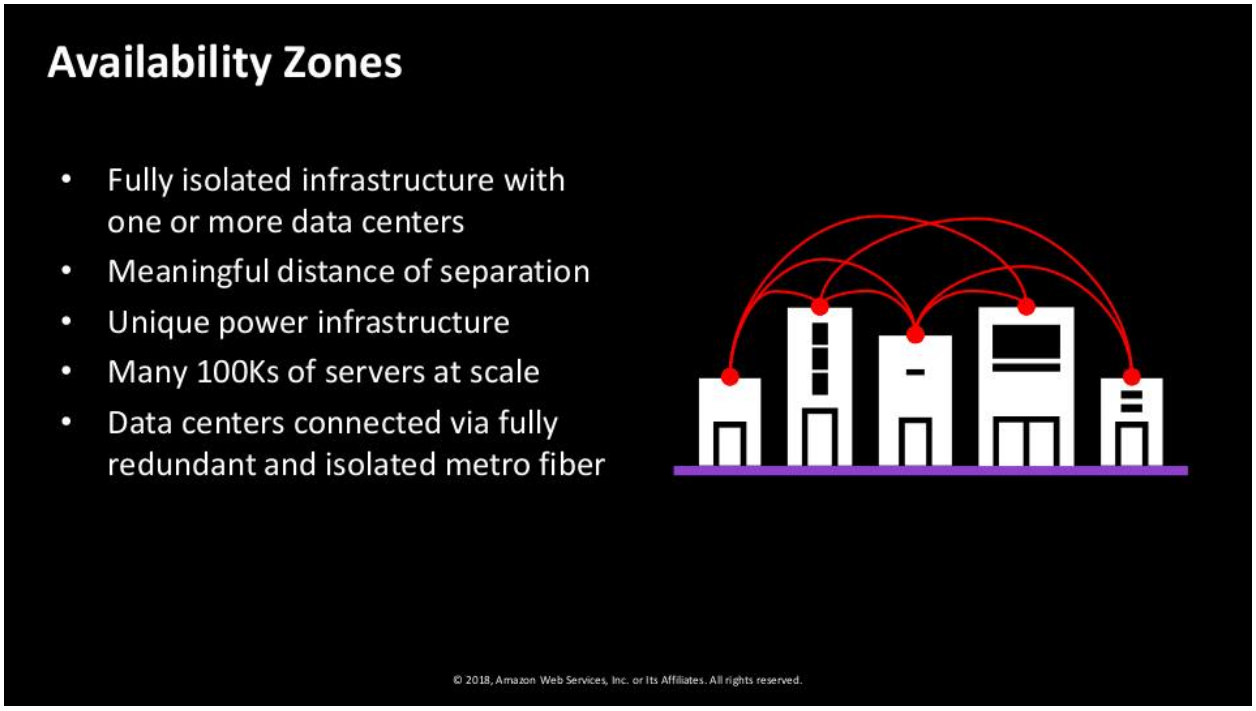


Expansion 2016-2018

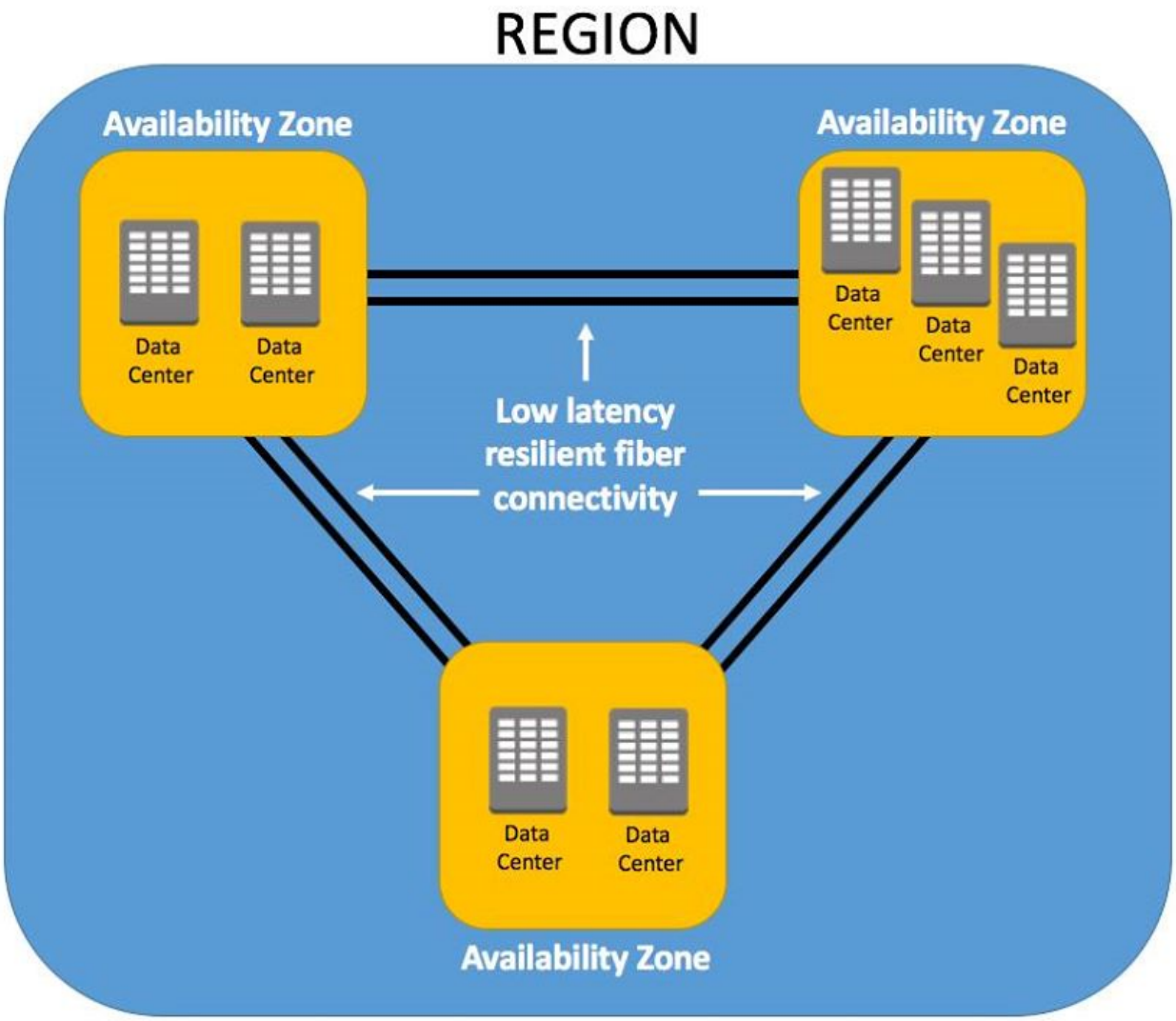
1.2 Availability Zones (AZs)

Availability Zone ဆိုတာကတော့ ရိုးရိုးရှင်းရှင်းနဲ့အလွယ်ပြောရရင် Region တစ်ခုဆီမှာ ရှိတဲ့ Physically Separated Data Centers တွေပါပဲ။ အပေါ်မှာပြောခဲ့သလိုပဲ Region တစ်ခုမှာ Availability

Zone လို့ခေါ်တဲ့ Data Centers တွေရှိကြပါတယ်။ သို့ပေမယ့် Availability Zone တစ်ခုကို Physical Data Center တစ်ခုဆိုပြီးလည်းသတ်မှတ်လို့မရပါဘူး။ ဆိုလိုခြင်တာကတော့ Availability Zone တစ်ခုမှာ Physical Data Center တစ်ခုကနေ အများဆုံး ၅ခုထိရှိတတ်ပါတယ်။ Standard အနေနဲ့ကတော့ Region တစ်ခုမှာ AZs ၃ ခုစီထားရှိမယ်ဆိုပေမယ့် တစ်ချို့သော Region တွေမှာ AZs ၂ ခုပဲရှိမျိုးလည်းရှိပါတယ်။ Regions မှာရှိတဲ့ AZs တွေအချင်းချင်းကို Low-Latency, High Throughput and Highly Redundant Networking, Highly Available, Fault Tolerant, Scalable infrastructure design တွေနဲ့ Well-Architect ပြုလုပ်ပြီး စနစ်တကျ တည်ဆောက်ထားတာပဲဖြစ်ပါတယ်။ Region တစ်ခုမှာ Availability Zone တွေအများအပြားရှိခြင်းကြောင့် Disaster Recovery (DR)အတွက်လည်းအနည်းနဲ့အများ ကာကွယ်ပြီးသားဖြစ်ပါတယ်။



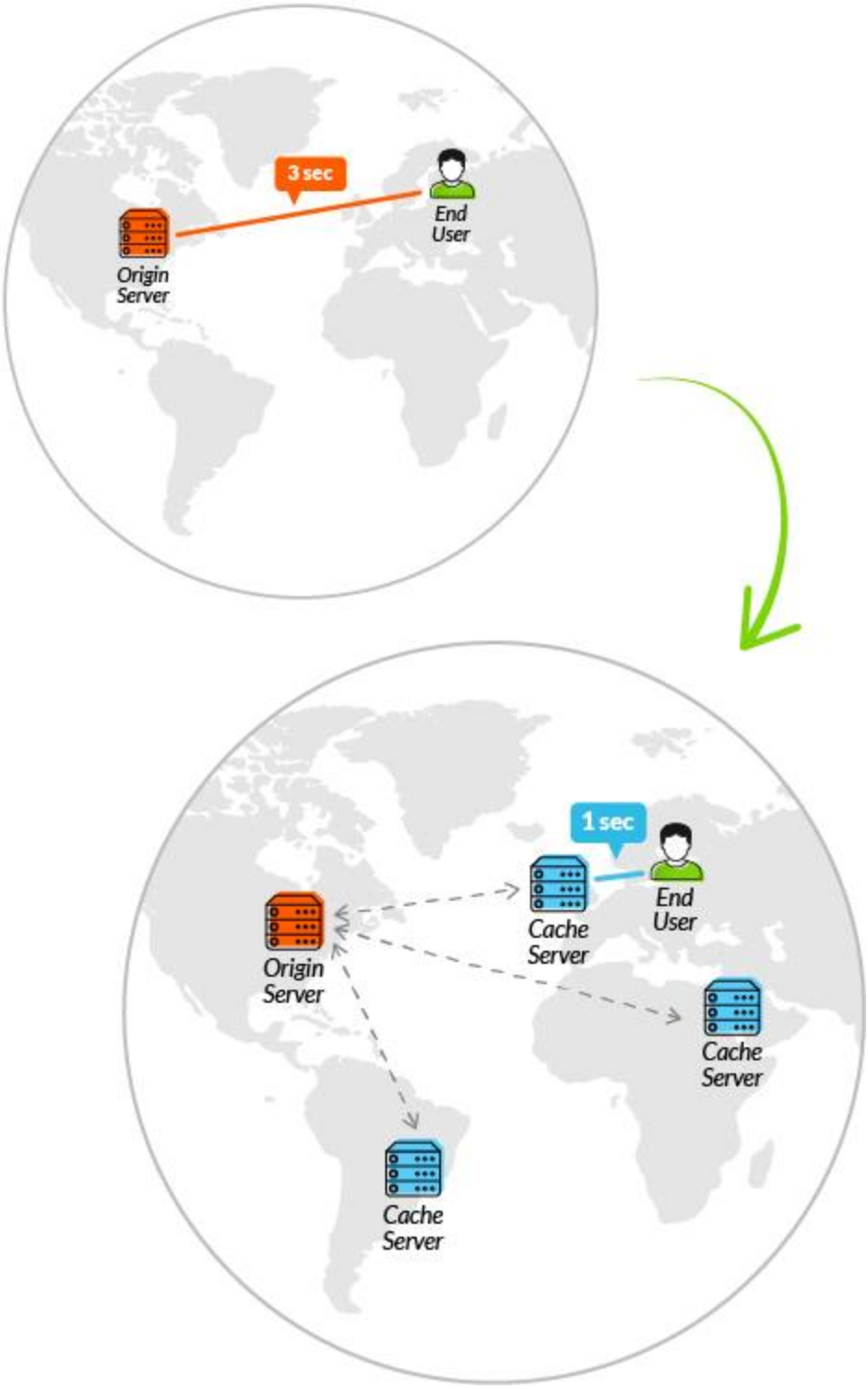
Availability Zones



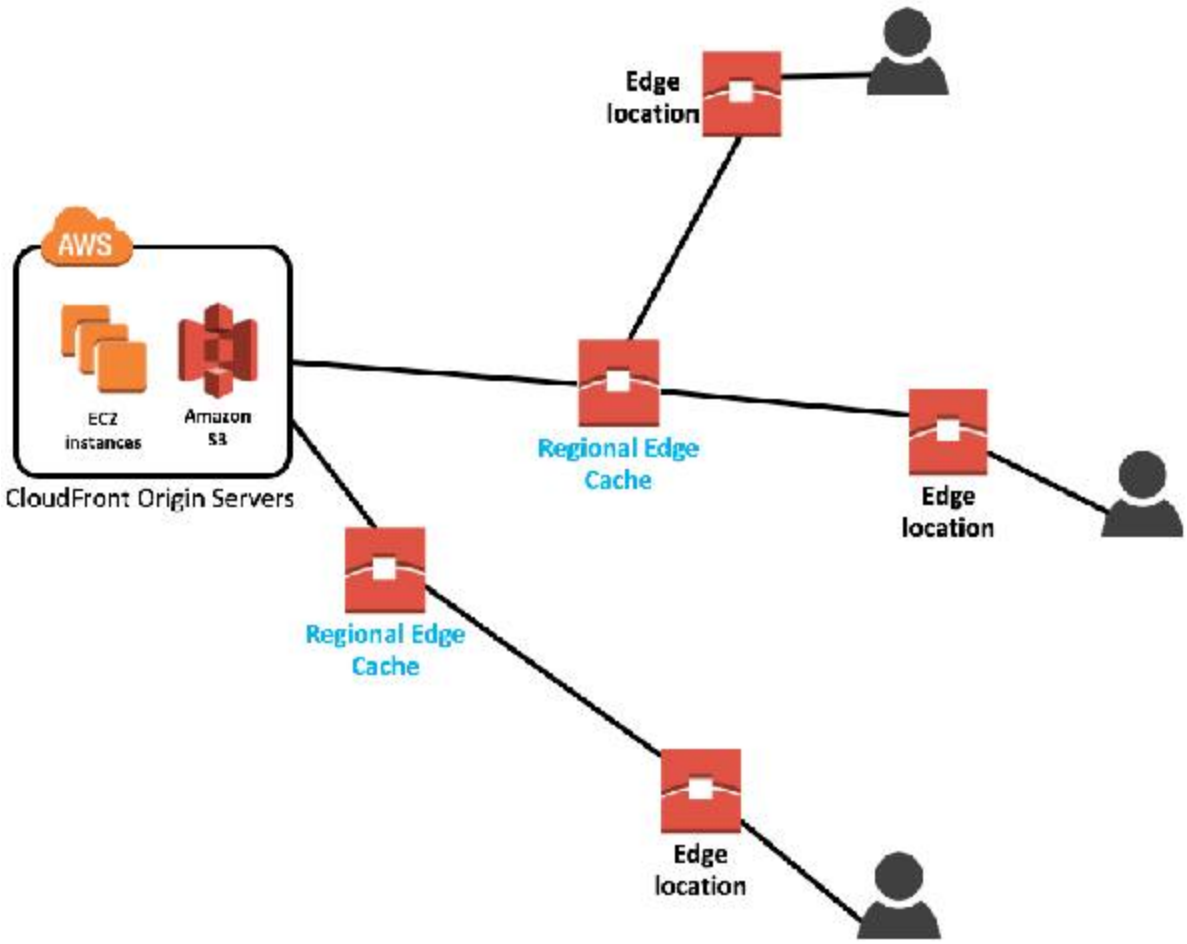
Region and Availability Zone Explained

1.3 Edge Locations

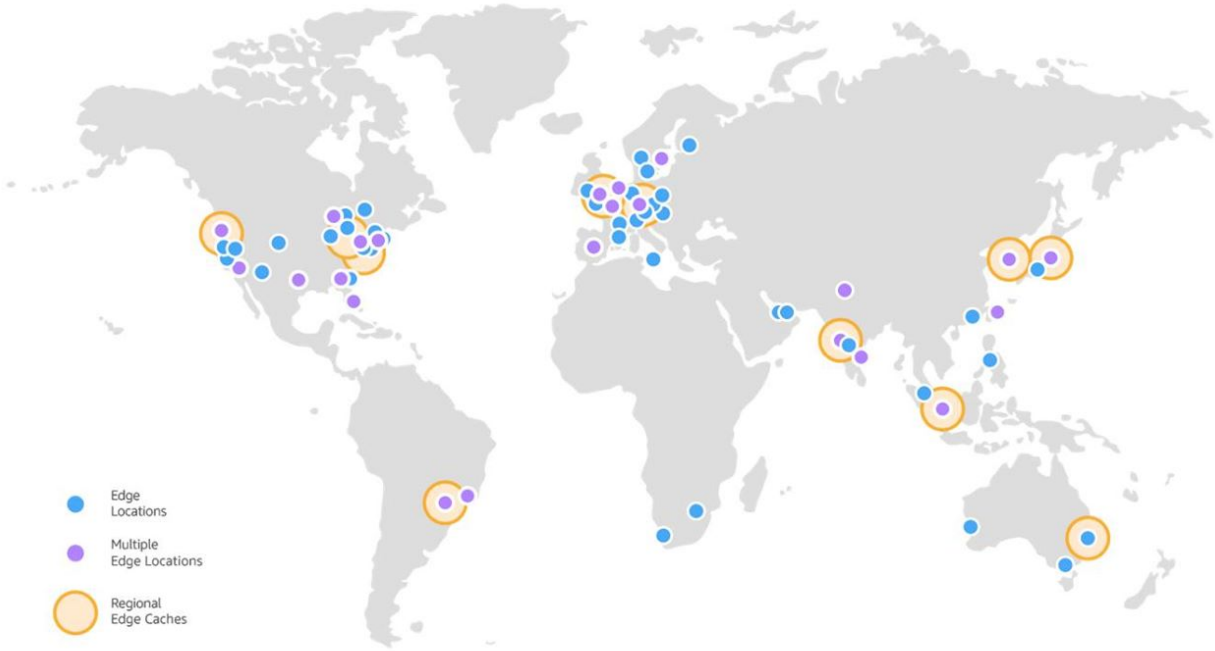
Edge Locations ဆိုတာကတော့ AWS က Sevice တစ်ချို့ (CloudFront, Lambda@Edge) တို့ကို User တွေအနေနဲ့အသုံးပြုရာမှာပိုမိုမြန်ဆန်ဆောင်ပြုလုပ်ပေးတဲ့ Global Content Delivery Network (CDN) EndPoints တွေပဲဖြစ်ပါတယ်။ အများသိကြတဲ့ အတိုင်းပဲ User တွေအနေနဲ့ ပုံမှန်အတိုင်း Content Delivery Network (CDN) မသုံးပဲ United State မှာရှိတဲ့ မိမိတို့ Server ကို တိုက်ရိုက် access လုပ်မယ်ဆိုရင် Latency အရ စက္ကန့်အနည်းနဲ့အများ ပိုကြာနိုင်ပြီးအဝိုင်းလေးလည်နေမှာပါ။ Edge Locations တွေက User အနေနဲ့ Access လုပ်လိုက်တဲ့ Content Data တွေကို Low-latency နဲ့ access ရအောင်လုပ်ပေးမှာပဲဖြစ်ပါတယ်။



How Content Delivery Network (CDN) works



Edge Locations explained



Point of Presence (PoPs)

Edge Location တွေအနေနဲ့လည်း Geographic Region တစ်ခုမှာ အနည်းဆုံး ၂ ခုဆီရှိတတ်ကြပါတယ်။ Global အတိုင်းအတာအနေနဲ့ Point of Presence (PoPs) ၁၅၀ (139 Edge Locations and 11 Regional Edge Caches) ရှိပါတယ်။ ဒီလောက်ဆိုရင်တော့ AWS ရဲ့ Global Infrastructure ကဘယ်လို ဒီဇိုင်းကာလာအနေအထားရှိတယ်ဆိုတာ
အားလုံးပဲနားလည်သဘောပေါက်လိမ့်မယ်လို့ယူဆမိပါတယ်ခင်ဗျာ။ နောက်ထပ်နောက်ထပ်သော စိတ်ဝင်စားစရာ အကြောင်းအရာကောင်းတွေကို ရေးသားသွားဦးမှာမို့ ဆက်လက်အားပေးကြပါဦးခင်ဗျာ။ Stay Tuned !!! Have a nice day

ကြိုးစားပါဦးမည်၊ အားလုံးကိုကျေးဇူးတင်လျက်

A brief History of AWS

AWS Introduction

နိုင်ငံတကာတွင် Cloud Computing ရေစီးကြောင်းတိုးတက်လာသည်နှင့် တပြိုင်နက် Cloud Computing စနစ်ကို ထောက်ပံ့ပေးနေသော provider များပေါ်ထွက်လာခဲ့ပါသည်။ Amazon, Google, Microsoft ကဲ့သို့သော Company ကြီးများနှင့်အတူ IBM, Alibaba, Century Link, Rackspace အစရှိသော Company ကြီးများမှလဲ Cloud Platforms များတည်ဆောက်ပြီး Service Provider အနေဖြင့် ရပ်တည်လျက်ရှိကြပါသည်။ ထို Company ကြီးများအနက် Amazon မှ AWS Cloud သည် 2017 ခုနှစ် Gartner Survey အရ Market Leader Cloud Provider အဖြစ်ထိပ်ဆုံးမှ ရပ်တည်လျှက်ရှိပြီး Public

Cloud Service အားအဓိက support ပေးလျှက်ရှိပါသည်။ AWS Cloud သည် 2017 စစ်တမ်းများအရ 17.46 Billion Annual Revenue အဖြစ် ဝင်ငွေအများဆုံး Cloud provider အဖြစ်ရပ်တည်နေပါသည်။ အထူးသဖြင့် လက်ရှိ AWS သည် Public Cloud Computing Provider ဖြစ် IAAS, PAAS, SAAS အစရှိသော Cloud Service များကိုထောက်ပံ့ပေးလျှက်ရှိသော Provider တစ်ခုအဖြစ်သာမက Hybrid Cloud Setup များကို Support ပေးနေသော Service များရှိသော Provider တစ်ခုလည်းဖြစ်ပါသည်။

AWS သမိုင်းကြောင်းများအား ပြန်ကြည့်ရမည်ဆိုလျှင် ကနဦး AWS Platform ကို 2002 July တွင် Service အနည်းငယ်မျှဖြင့် စတင်ခဲ့ပြီး 2003 နှောင်းပြီးကာလတွင် Chris Pinkam and Benjamin Black တို့မှစနစ်တကျ Reformulated ပြုလုပ်ခဲ့ပြီး 2004 November တွင် AWS ၏ပထမဦးဆုံး Service ဖြစ်သည့် SQS ကို Public Launch ပြုလုပ်ခဲ့ပါသည်။ ထို့နောက် AWS ၏ဒုတိယမြောက် Service

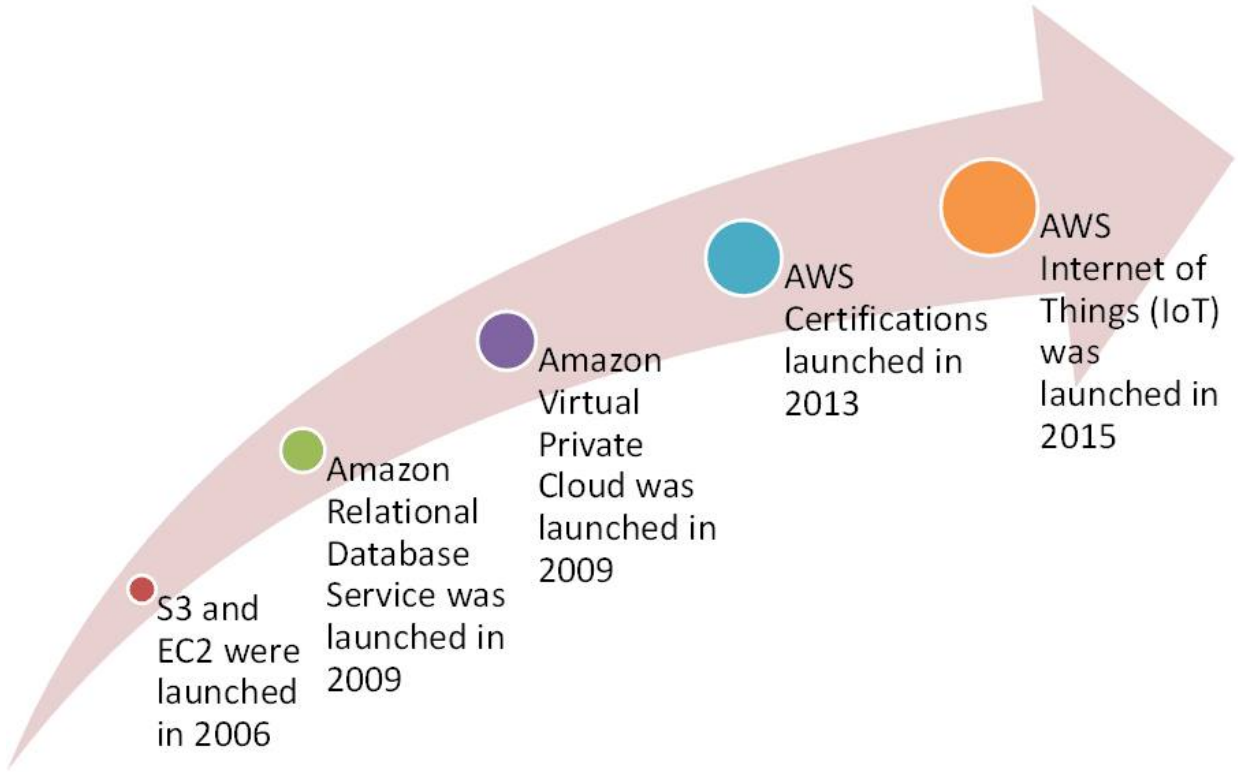
ဖြစ်သည့် Elastic Compute Cloud (EC2) ကို တောင်အာဖရိကနိုင်ငံ ကိတ်တောင်းမြို့တွင် Develop ပြုလုပ်ပြီး တရားဝင် Publice Relaunch အား 2006, March14 တွင်

(1) S3

(2) SQS

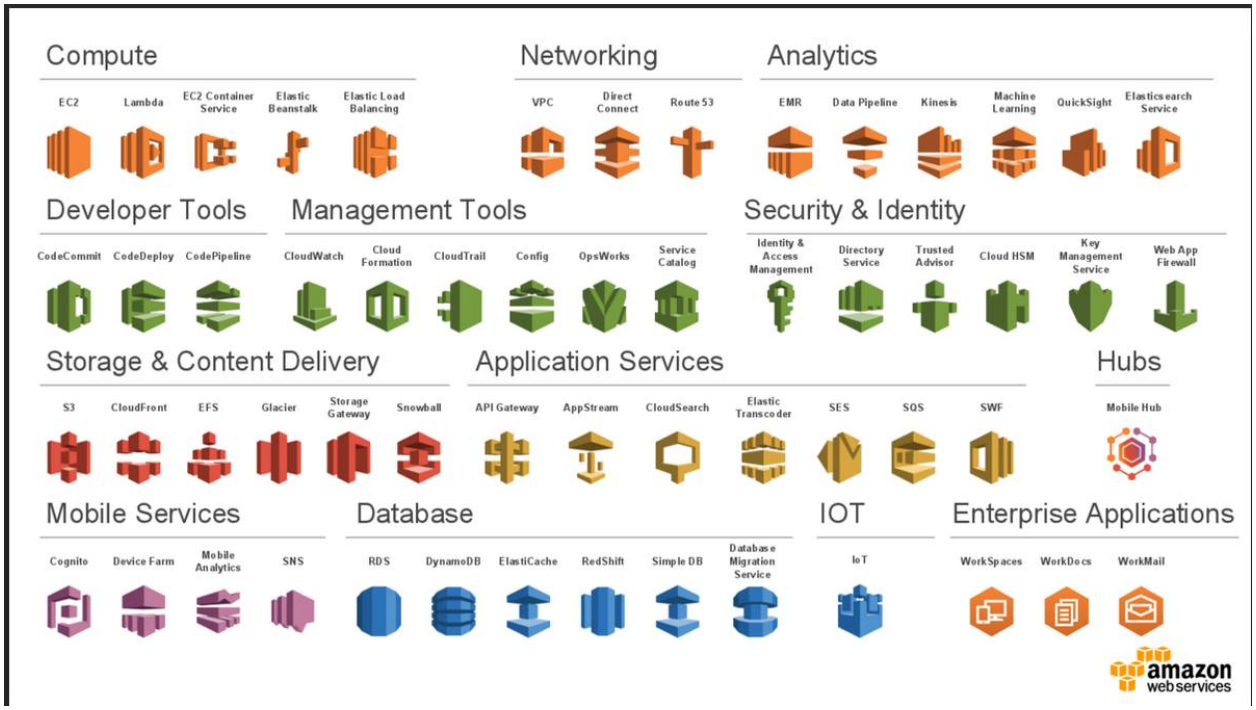
(3) EC2 အစရှိသော Initial Service (3) မျိုးဖြင့် စတင်ခဲ့ပါသည်။

Andy Jassy က တော့ 2006 တွင် Vice President အနေဖြင့် တာဝန်ထမ်းဆောင်ခဲ့ပြီး 2016 တွင် CEO အဖြစ်ယနေ့တိုင် ဆောင်ရွက်လျှက်ရှိပါသည်။ ယနေ့ခေတ် Market တွင် Service ပေါင်းတစ်ရာကျော်ဖြင့် Leading Cloud Computing Provider အဖြစ်ရပ်တည်နေပြီးဖြစ်ပါသည်။



ယနေ့ခေတ် AWS Core Services များတွင်အရေးကြီးပြီးအသုံးများသော Services အချို့အားလေ့လာကြည့်ရမည်ဆိုလျှင် Compute,Storage, Database, Networking, Security and Management များဖြစ်ကြပါသည်။ထို့အပြင် AWS တွင် အခြားသော Services များ၊ နောက်ဆုံးပေါ် Technology Services များလည်းရှိနေပါသေးသည်။ အထူးသတိပြုရမည်ဆိုလျှင် ထို Services အားလုံးကို တပြိုင်နက်ထဲ အသုံးပြုနိုင်ဖို့အတွက် ခက်ခဲကောင်းခက်ခဲနိုင်ပါသည်။ မိမိ Organization, Company ၏ လိုအပ်ချက်အပေါ်မူတည်ပြီး အသုံးပြုသော Services အမျိုးအစားများလည်း ကွဲပြားသွားနိုင်ပါသည်။ ဆိုလိုသည်မှာ Services အားလုံးကိုတပြိုင်တည်းနှင့် အသုံးပြုရန် မလိုအပ်ပါ။

ယေဘူယျအားဖြင့် Services ၏ အလုပ်လုပ်ပုံ၊အသုံးပြုပုံ များကိုသိရှိရန် သာလိုအပ်ပါသည်။ မိမိအသုံးပြုမည့် Services ၏သဘောတရားများကို နားလည်ပါက Services ၏ Documentation များကို အခြေခံ၍ Implement ပြုလုပ်နိုင်ပါသည်။ သို့ရာတွင် AWS Cloud ကိုစတင်အသုံးပြုမည်ဆိုပါက AWS ၏အဓိကအသုံးများဆုံးဖြစ်တဲ့ Core Services များအား သိရှိနားလည်ရန်တော့လိုအပ်ပါသည်။



AWS ၏ အဓိက Core Services များကတော့

Compute EC2 (Elastic Compute Cloud) ECS (Elastic Container Service)

Storage S3 (Simple Storage Service) EBS (Elastic Block Storage) EFS (Elastic File System)

Database RDS DynamoDB Elasticache Redshift

Networking VPC (Virtual Private Cloud) Route 53

Security IAM (Identity and Access Management)

Management Cloud Watch Cloud Trail Config

အစရှိသော Service တစ်ခုချင်း၏ ရည်ရွယ်ချက် အလုပ်လုပ်ဆောင်ပုံများ ကွဲပြားကြပါသည်။

ကျွန်တော်တို့ နောက်ထပ်ပို့စ်များတွင် တစ်ခုချင်းစီ၏ အလုပ်လုပ်ပုံ၊ ရည်ရွယ်ချက်များကို အချိန်ရရင် ရသလိုတင်ပေးသွားပါမည်။ အခုလောလောဆယ်တော့ AWS ကဘာလဲ? သမိုင်းကြောင်းနှင့် သူ့ရဲ့

Ecosystem/Core Services တွေက ဘာတွေရှိလဲ အစရှိတဲ့ အကြောင်းအရာများကို ဤစာစုကလေးက ပေးနိုင်လိမ့်မယ်လို့ ကျွန်တော်မျှော်လင့်မိပါတယ်။

Cloud Computing 101

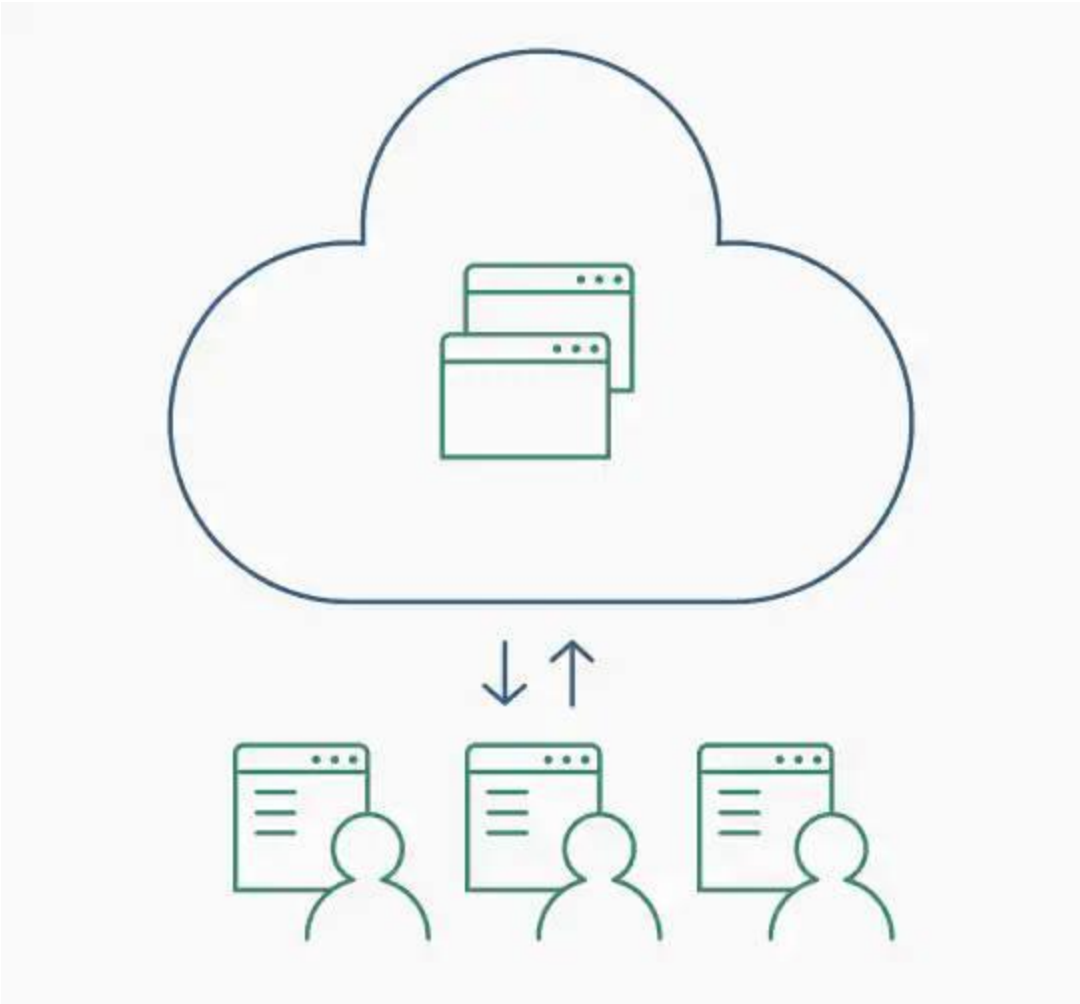
Introduction to Cloud Computing

Cloud Computing လို့ပြောလိုက်တာနဲ့ ကျွန်တော်တို့အရင်ဆုံး တိမ်ပုံစံလေးကို မျက်မှန်းတန်းမိမှာအသေအချာပါပဲ။ ပထမဦးစွာ ယခင်က Cloud Computing တွေမသုံးခင်က ကျွန်တော်တို့ဟာ မိမိ Organization သိုမဟုတ် Company အတွက် ကိုယ်ပိုင် Data Center ကို မိမိတို့ကိုယ်တိုင်တည်ဆောက်ကြရပါတယ်။ ဒါတင်မကသေးပါဘူး Infrastructure တစ်ခုလုံးတည်ဆောက်ပြီးသွားရင်တောင်မှ ထိုအတွက်လိုအပ်တဲ့ အလုပ်သမားဌားရမ်းခ၊မီတာခ အစရှိသဖြင့် additional cost တွေပါမလွဲမသွေကုန်ကြမှာပါ။ တစ်ခါတစ်လေ အသုံးမလိုပဲ ပိုနေတဲ့ Resources တွေ၊သဘာဝဘေးအရေးပေါ် ကြုံကြုံကြုံကလာတာမျိုးတွေ၊ Emergency Resource လိုအပ်တာမျိုး စသည့်အခြေအနေတွေကို Organization တွေနေနဲ့ သီးသန့်အချိန်ပေးခေါင်းရှုပ်မခံကြတော့ပါဘူး။ အဲဒါနဲ့နောက်ပိုင်းမှာ ရွေးချယ်စရာများစွာရှိတဲ့ Cloud Service Providers တွေကိုရွေးချယ်လာကြပါတယ်။ Cloud Computing ဆိုတာကတော့ Computing Services တွေဖြစ်ကြတဲ့ Servers,Storages,Databases,Networking,Applications,Analytics နဲ့ များစွာသော IT နှင့်ဆက်စပ် services များကို အင်တာနက်ပေါ်မှတစ်ဆင့် **Pay-As-You-Go (မိမိကြိုက်သလောက်သုံးကျသလောက်ရှင်း)** စနစ်နဲ့ ထောက်ပံ့ပေးတာမျိုးဖြစ်ပါတယ်။ တစ်နည်းအားဖြင့် ယနေ့အသုံးပြုနေကြတဲ့ သမာရိုးကျ Data Center တွေမှာရနိုင်တဲ့ Hardware/Software Resources အတော်များများတွေကို Cloud Service Providers တွေရဲ့ Virtual Environment ပေါ်မှာ အချိန်ကုန်လူပင်ပန်းသက်သာစွာနဲ့ မိမိပိုက်ဆံပေးနိုင်ရင်ပေးနိုင်သလောက်ရနိုင် အသုံးချနိုင်တာမျိုးပဲဖြစ်ပါတယ်။

လက်ရှိအချိန်မှာတော့ cloud အမျိုးအစားကို အသုံးချတဲ့ပုံစံအဓိက (၃) မျိုးတွေ့နိုင်ပါတယ်။ယနေ့အချိန်မှာတော့ နောက်ထပ်အမျိုးအစားငယ်လေးတွေလည်းရှိလာပါပြီ။

1. Public Cloud

Public Cloud ကတော့ infrastructure ကိုသက်ဆိုင်ရာ third-party cloud service providers (AWS, GOOGLE, IBM, ORACLE) တွေကပိုင်ဆိုင်ကြတာဖြစ်ပြီး user တွေလိုအပ်တဲ့ virtual resources တွေကို အင်တာနက်ပေါ်မှတစ်ဆင့် delivery လုပ်ပါတယ်။ User တွေအနေနဲ့ မိမိတို့ web browser မှတစ်ဆင့်ဝင်ရောက်ပြီး manage လုပ်ဖို့အတွက် access လုပ်ကြရပါတယ်။ တစ်ချို့သော Organization တွေမှာတော့ on-premises မှ public cloud ကိုပြောင်းလဲအသုံးပြုလာကြပါတယ်။



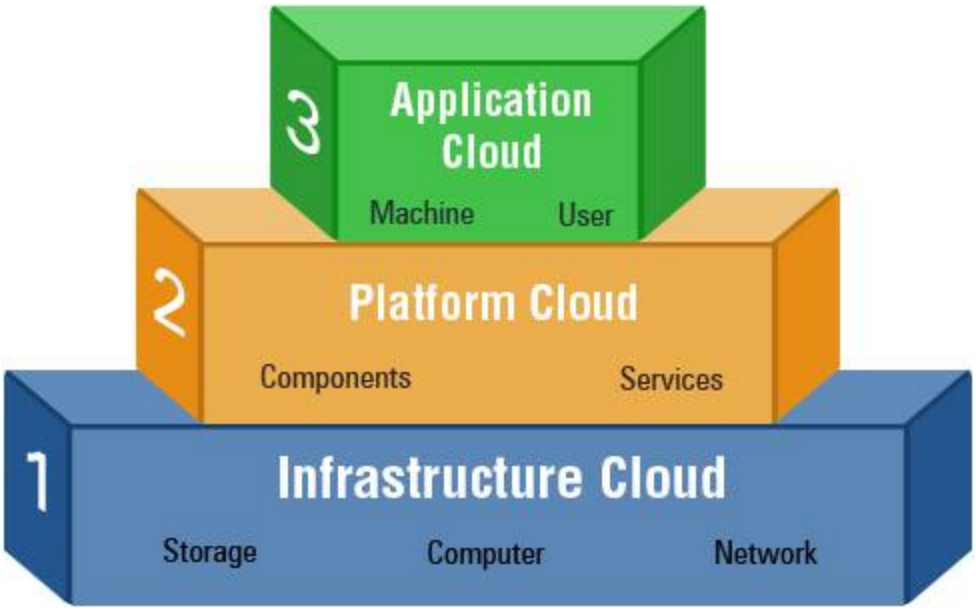
2. Private Cloud or On-premises

Private Cloud ကတော့ infrastructure ကို single oragnization/business တစ်ခုတည်းကပဲသီးသန့်ပိုင်ဆိုင်အသုံးပြုလို့ရပါတယ်။ Public cloud မှာလိုမျိုး မည်သူမဆိုအသုံးပြုနိုင်မယ်ဆိုတာမျိုးမဟုတ်ပါဘူး။ Dedicated ဖြစ်ပါတယ်။ အရင်ကတော့ မြန်မာနိုင်ငံက company တစ်ချို့မှာ on-premises ကိုသာသုံးဆွဲနေကြမယ်လို့လဲယူဆနိုင်ပါတယ်။ Data Center တွေတော့ကိုယ်ပိုင်ရုံးတွင်း environment သို့မဟုတ် colocation center တွေမှာ host လုပ်လေ့ရှိကြပါတယ်။ Private cloud ကိုတော့ information sensitive ဖြစ်တဲ့ government agencies, financial institutions, mid to large-size organization တွေမှာသုံးလေ့ရှိကြပါတယ်။တစ်နည်းအားဖြင့် ကိုယ့် information/data တွေကို အပြင်ကိုပေါက်ကြားမှာကိုမလိုလားတဲ့ organization အနေနဲ့အများဆုံးရွေးချယ်ကြပါတယ်။

3.Hybrid Cloud

Hybrid Cloud ကတော့ ရှင်းပါတယ်။ ရိုးရိုးလေးပဲတွေးကြည့်တာနဲ့တင် ဘယ်လိုအမျိုးအစားလဲဆိုတာပေါ်လွင်ပါတယ်။ သူကတော့ private ရော public ပါ ၂ခုလုံးပေါင်းထားတဲ့ cloud အမျိုးအစားဖြစ်ပါတယ်။ ဥပမာအနေနဲ့ဆိုရင်တော့ ကျွန်တော်တို့ service ပေးမဲ့ lower-security လောက်ပဲလိုအပ်တဲ့ application တစ်ခုမှာ infrastructure အနေနဲ့ resource တွေကို public cloud မှာထားမယ်၊ အရေးကြီးတဲ့ Business-critical operation တွေလိုမျိုးအခြေအနေမျိုးမှာဆိုရင် resource တွေကို private cloud မှာထားမယ်စသဖြင့် စီစဉ်ထားနိုင်ပါတယ်။ Public ရော Private နှစ်မျိုးစလုံးမှာရနိုင်မဲ့ features တွေကို ကိုယ်ပိုင် Organization မှာလိုချင်တာမျိုးဆိုရင်လည်း Hybrid cloud ကိုရွေးချယ်ကြပါတယ်။

Cloud Computing Models



Cloud computing models တွေအနေနဲ့မှာလည်း အဓိက(၃)မျိုးတွေ့နိုင်သေးတယ်။ နာမည်ကြီး Cloud Service Providers တော်တော်များများမှာတော့ models (၃)မျိုးစလုံးကို အထောက်အပံ့ပေးနိုင်ကြပါတယ်။

1.Infrastructure as a Service (IaaS)

ဒီ IaaS model အမျိုးအစားကတော့ ကျွန်တော်တို့ User/Organization တွေကလိုအပ်တဲ့ computing infrastructure တွေဖြစ်တဲ့ Server,Block & file-based storages,Networking features,LoadBalancers,IP Addresses တို့ကို အထောက်အပံ့ပေးနိုင်ပါတယ်။ ဥပမာအနေနဲ့ဆိုရင်

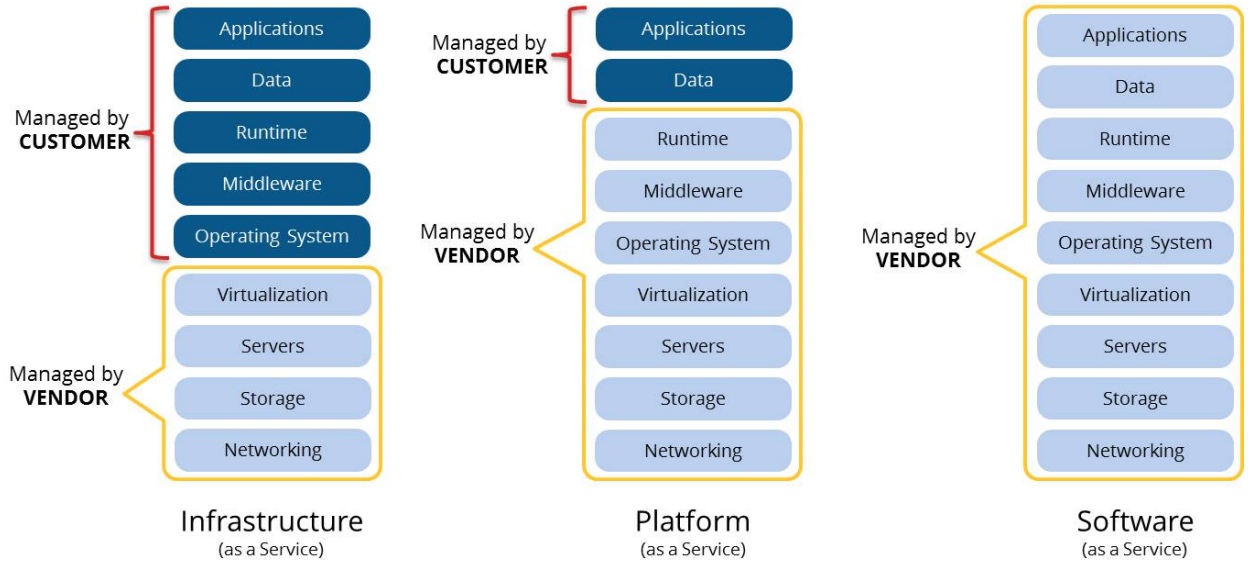
Virtual Private Server (VPS) တစ်လုံးကို Digital Ocean ကနေဝယ်ယူလိုက်တယ်ဆိုရင် ကျွန်တော်တို့အပြင်မှာရင်နှီးနေကြဖြစ်တဲ့ Server ကိုကိုယ်တိုင် OS တင်၊ Network ချိတ်လုပ်ရမဲ့အရာတွေကိုလုပ်စရာမလိုတော့ပါဘူး။ အဆင်သင့် OSတင်ပြီးသား internet connection လည်း ကောင်းကောင်းနဲ့အလွယ်တကူပဲ VPS ကို access လုပ်နိုင်ပါပြီ။ ဈေးကွက်ထဲမှာတွေ့နိုင်တဲ IaaS provider အချို့တွေကတော့ Amazon Web Service(AWS),Digital Ocean(DO),Microsoft Azure တို့ပဲဖြစ်ပါတယ်။ Iaas ရဲ့ ကောင်းကွက်ဆိုးကွက်တွေကိုတော့ နောက်မှဆက်လက်ရေးသားသွားပါဦးမယ်။

2.Platform as a Service(PaaS)

PaaS model မှာတော့ အပေါ်မှာရှင်းပြပြီးဖြစ်တဲ့ IaaS model အမျိုးအစားနဲ့မတူတော့ပါဘူး။ Platform as a Service ဆိုတဲ့အတိုင်းပဲ computing platform တွေဖြစ်တဲ့ operating system,programming language execution environment,database,webserver တွေကိုအလွယ်တကူအဆင့်သင်ယူသုံးနိုင်တဲ့အဆင့်ထိ service ပေးပါဖြစ်ပါတယ်။ ကျွန်တော်တို့အနေနဲ့ web application တစ်ခုကို develop လုပ်မယ်ဆိုရင် application တွေ manually လိုက်လံရှာဖွေ install/configure လုပ်နေစရာမလိုတော့ပါဘူး။ ဒါ့အပြင် Operating System တစ်ခုခုဖြစ်မှာတို့ storage capacity မလောက်မှာတို့အတွက်လည်းပူစရာမလိုတော့ပါဘူး။ မိမိအနေနဲ့ ဘယ် database အမျိုးအစားကိုသုံးမယ် webserver ဆိုလည်း apache သုံးမယ်ဆိုတာကိုရွေးပေးလိုက်ရုံနဲ့ အလွယ်တကူ instant သုံးနိုင်တဲ့အဆင့်ထိရောက်အောင် providers တွေကဆောင်ရွက်ပေးသွားမှာဖြစ်ပါတယ်။

3.Software as a Service(SaaS)

Software as a Service model ကတော့ကျွန်တော်တို့နဲ့တော်တော်လေးရင်နှီးကျွမ်းဝင်ပြီးသား model အမျိုးအစားထဲကပဲဖြစ်ပါတယ်။ သူ့မှာတော့ IaaS တို့ PaaS တို့လိုမျိုး Infrastructure ရော Platform ကိုပါ maintain လုပ်စရာမလို့ပဲအသင့်သုံးနိုင်တဲ့ model မျိုးဖြစ်ပါတယ်။ ဥပမာအနေနဲ့ Gmail ပဲထားပါတော့ဗျာ၊ ကျွန်တော်တို့လို သာမန်လူတွေအနေနဲ့ Gmail ကိုအသုံးပြုဖို့ဘယ် server,database,network ကိုမှ configure/maintain လုပ်စရာမလိုပါဘူး။ Email Service အနေနဲ့ Gmail Application ကို Google ကတာဝန်ယူပြီးအားလုံးအလွယ်တကူသုံးလို့ရအောင်ဆောင်ရွက်ပေးထားလို့ပဲဖြစ်ပါတယ်။ ထိုနည်းတူ facebook,twitter,Google+ တွေကလည်းဒီသဘောတရားပါပဲ။ User တွေအလွယ်သုံးနိုင်အောင် Software as a Service အနေနဲ့ provide လုပ်ပေးထားလို့ပဲဖြစ်ပါတယ်။



အပေါ်ကပုံကိုကြည့်ပြီးတော့ Cloud Computing Models
တွေကိုသေချာကွဲကွဲပြားပြားမျက်စိထဲမြင်ယောင်မယ်လို့မျှော်လင့်ပါတယ်ခင်ဗျာ။

နောက်ထပ် Topic တွေမှာလည်းယခုထပ်ပိုတဲ့အကြောင်းအရာတွေကို
ဆက်လက်ရေးသားသွားပါဦးမယ်ခင်ဗျာ။

Photo Ref: Google

Elastic Compute Cloud (or) EC2 - Episode (1)

Elastic Compute Cloud Introduction

ဒီနေ့မှာတော့ AWS မှာရှိတဲ့ Service တွေထဲကမှ တစ်ခုအပါအဝင်ဖြစ်ပြီး လူသုံးအများဆုံး Core Service တစ်ခုလည်းဖြစ်တဲ့ EC2 (Elastic Compute Cloud) Instance အကြောင်းကိုပြောပြပေးသွားမှာဖြစ်ပါတယ်။ ဆိုတော့ကာ ကျွန်တော်တို့ကြားဖူးနေတဲ့ AWS က EC2 Instance ဆိုတာဘယ်လို Service လဲဆိုရင်တော့ဖြင့် အနီးစပ်ဆုံးပြောရင် Virtual Private Server (VPS) တွေလိုပါပဲ။ EC2 ကအဓိကအားဖြင့် Cloud Compute ပိုင်းကိုလုပ်ဆောင်ပါတယ်။ Compute ဆိုတဲ့ထဲမှာအများကြီးပါဝင်ပါတယ်။ လွယ်လွယ်ပြောရရင် ကျွန်တော်တို့သုံးချင်တဲ့ Web Server, Application Server, Database Server စတာတွေအတွက် Cloud ပေါ်မှာတင်ပြီးအသုံးပြုလို့ရနိုင်အောင်လုပ်ပေးတယ်ပေါ့ဗျာ။ EC2 Instance တွေဟာ သေချာမွန်းမံထားတဲ့ Xen Virtualization ပေါ်မှာအခြေခံထားတာဖြစ်ပြီး 2017 နောက်ပိုင်း Instance Class တစ်ချို့ကိုတော့ KVM Based Virtualization (Nitro) ပေါ်မှာအခြေခံထားတာဖြစ်ပါတယ်။ EC2 (Elastic Compute Cloud) ပေါ်မှာ Linux Distro တော်တော်များများ၊ BSD ၊ Windows Operating System စတာတွေ Run လို့ရပါတယ်။On-Premise မှာ ကိုယ်တိုင် OS Install/Setup လုပ်ရမဲ့ အချိန်တွေကို Cloud ပေါ်မှာဆိုရင်တော့ မိနစ်ပိုင်းအတွင်းပြီးမှာဖြစ်တဲ့ အတွက်အချိန်ကုန်သက်သာစေပြီး Future Plan အတွက် Scaling (Horizontal/Vertical) ပြုလုပ်ရာမှာလည်းအလွယ်တကူပြုလုပ်နိုင်မှာဖြစ်ပါတယ်။

Amazon EC2 (Elastic Compute Cloud) တွင် Virtual Server အပြင် ၎င်းမှ Supporting ပေးထားသော Features များကိုမိတ်ဆက်ပေးပါရစေ။

Basics

- Instances and AMIs
- Regions and Availability Zones
- Instance Types
- Tags

Networking and Security

- Amazon EC2 Key Pairs
- Security Groups
- Elastic IP Addresses
- Amazon EC2 and Amazon VPC

Storage

- Amazon EBS
- Instance Store

Guide

- အရင်ဆုံး Naming အနေနဲ့ AWS မှာ EC2 ကို ***Instance*** လို့လည်းခေါ်ကြပါတယ်။
- Resources များအတွက် Multiple Physical Locations ခေါ် ***Regions, Availability Zones (AZs)***
- CPU, Memory, Storage, Networking Capacity ဘယ်လောက်သုံးမယ်ကိုရွေးချယ်နိုင်မယ့် ***Instance Types***
- Instance အတွင်း Additional Software များကိုအချိန်တိုအတွင်းရွေးချယ်ပြီး Install/Setup လုပ်နိုင်သော Pre-configure Template ခေါ် ***AMIs (Amazon Machine Images)***
- Instance များကို လုံခြုစွာ Remote Login ပြုလုပ်နိုင်ရန်အတွက် ***Key Pair (Public/Private Key)***
- Instance ကို Stop or Terminate လုပ်ရင်ပျက်သွားမဲ့ Temporary Storage Volume အတွက် ***Instance Store Volumes***
- Persistent/Permanent Storage Data Volume အတွက် ***Amazon Elastic Block Storages (Amazon EBS Volumes)***
- Instances များအတွက် ***Security Groups*** ခေါ် Firewall (Specify the protocols, ports, and source IP range)
- ***Elastic IP*** Addresses ခေါ် Instance များအတွက် Static Public IPv4 Addresses
- EC2 Instances များအတွက်လိုအပ်သော Virtual Network များကို AWS Cloud အတွင်း Logically Isolated ဖြစ်ရန်အတွက်သတ်မှတ်နိုင်သော ***Amazon Virtual Private Clouds (Amazon VPCs)***
- AWS အတွင်းရှိ Instances, Amazon Machine Images (AMI) နှင့် အခြားသော Resources များကို Manage လုပ်ရလွယ်ကူစေရန်အတွက် ***Tags***

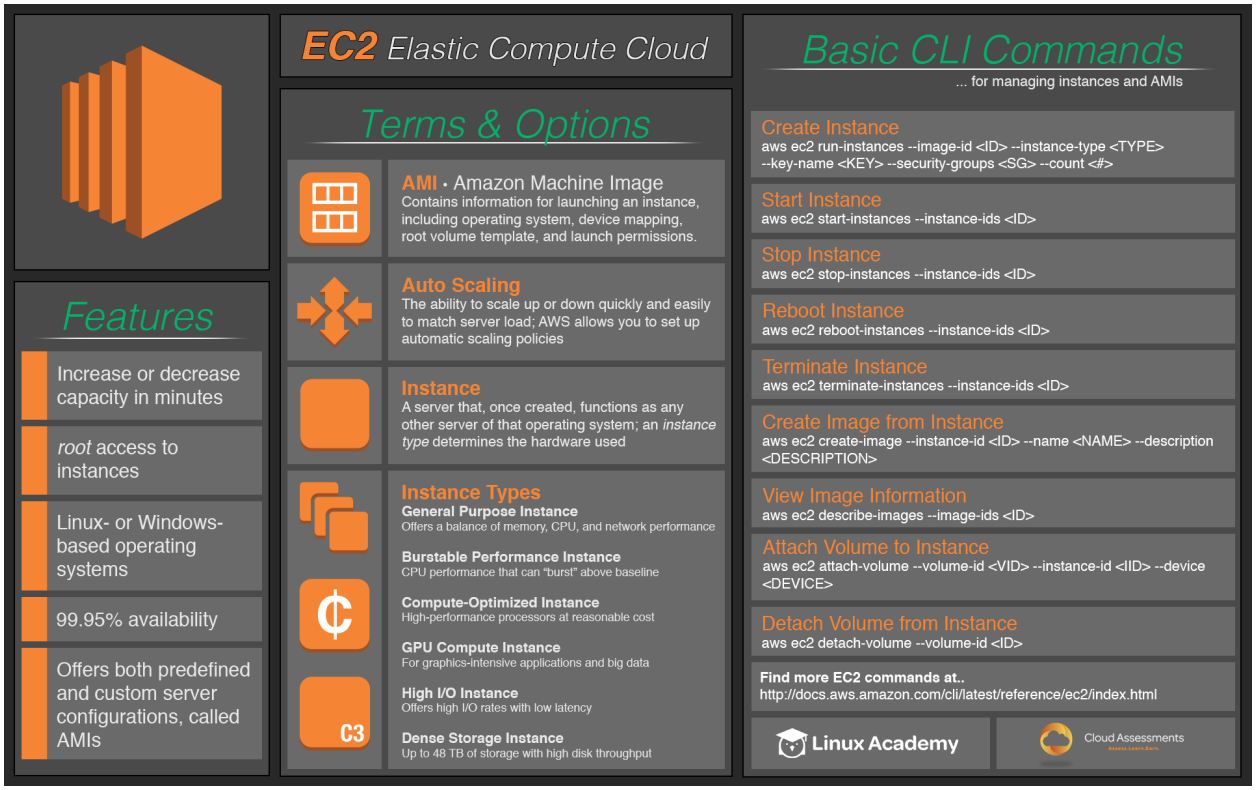
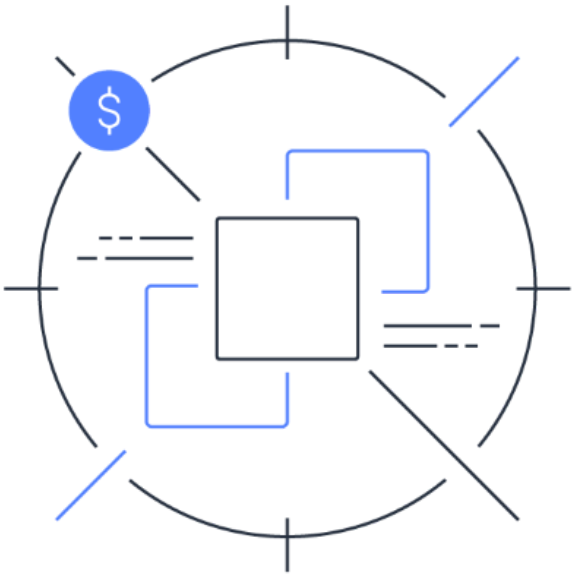


Photo Reference: Linux Academy

သည်လောက်ဆိုရင်တော့ **Amazon EC2** ရဲ့ အခြေခံ Concepts နဲ့ သူရဲ့နာမည်အခေါ်အဝေါ်လေးတွေကို အကြမ်းဖျင်းသဘောပေါက်လိမ့်မယ်ထင်ပါတယ်။ Episode (1) ပ်ရှိသေးတဲ့အတွက်နောက်ထပ် Episodes တွေမှာ Feature တစ်ခုချင်းစီရဲ့ Use Case နဲ့ Details လေးတွေရေးသားသွားဦးမှာမို့ ဆက်လက်အားပေးကြပါဦး ခင်ဗျာ။

Reduce cloud compute cost with EC2 Spot Instance

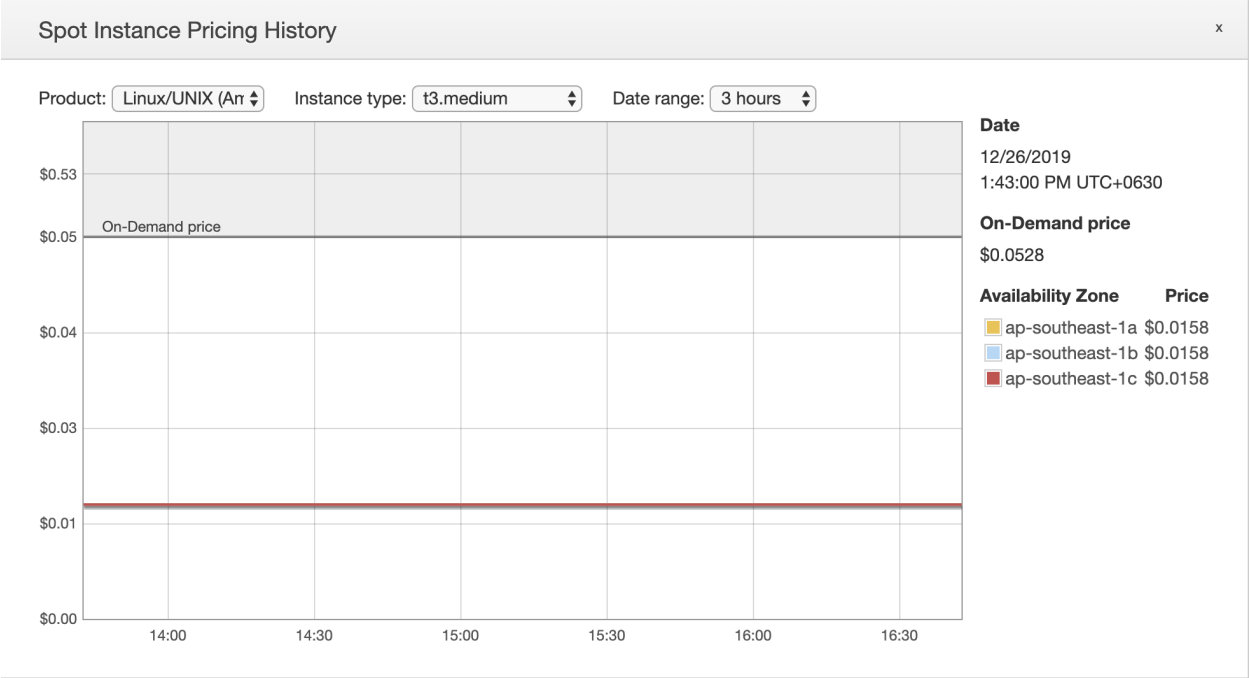
AWS Spot Instance Introduction with use case



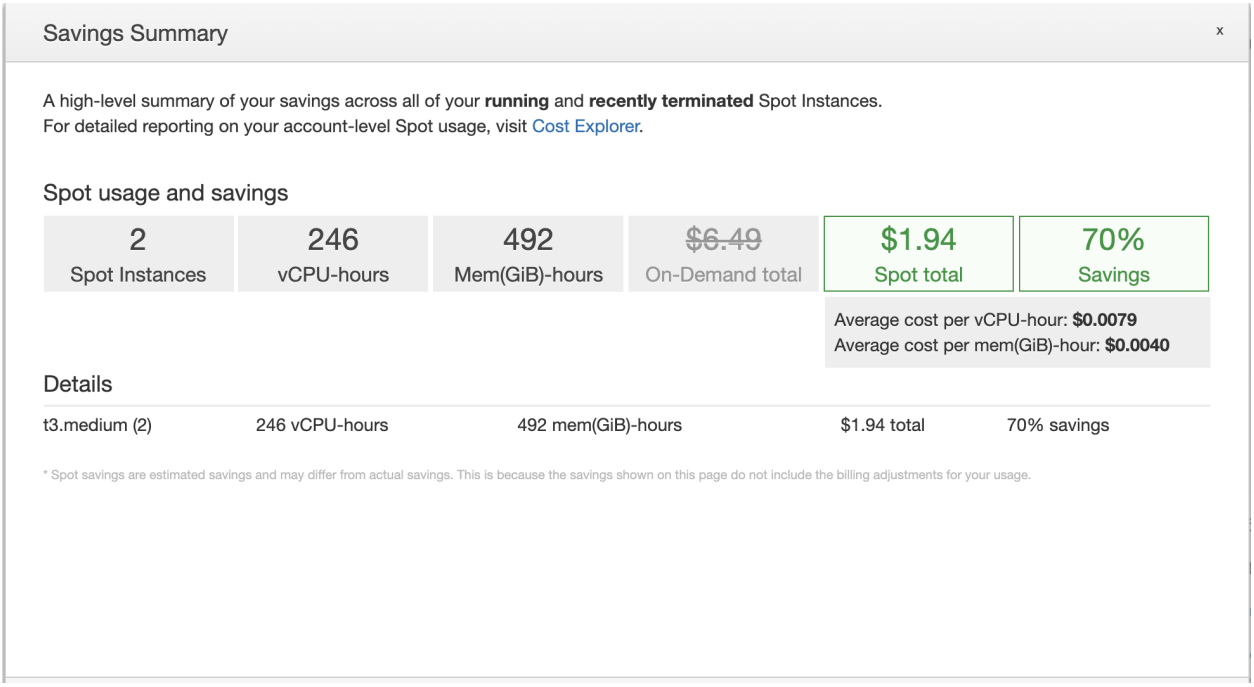
EC2 Spot Instance လို့ ပြောလိုက်တာနဲ့ တချို့က AWS ရဲ့ Pricing Model တစ်မျိုး or အသုံးမလို ပိုနေတဲ့ Compute Capacity တစ်ခု ခဏလေးသုံး အချိန်မရွေးပြန်ယူမှာ ဆိုတဲ့ အတွေးတွေနဲ့ သုံးဖို့ ဒီဟာဖြစ်နေကြတာတွေ့ရပါတယ် အမှန်တကယ်တော့ Spot instance က သူ့ နေရာသူ Usecase နဲ့ မှန်မှန်ကန်ကန်အသုံးပြု အသုံးချမယ် ဆို မိမိ Infrastructure ရဲ့ compute cost လျော့ချနိုင်မည့်အပြင်မှာ Application HA ပါ တခါ တည်း Setup ချ လို့ ရပါတယ်။ ကဲ ဒါဆို Spot Instance ကို ဘယ်လိုသုံးမလဲ ဘယ်နေရာမှာသုံးမလဲဆိုတာ ကြည့်လိုက်ရအောင်..အဲ့တာတွေ မပြောခင် Spot Instance က ဘာလဲ ဘာလို့ အရမ်းစျေးသက်သာလဲ ဆိုတာ တချက် သုံးသပ်ကြည့်ရအောင်ဗျာ ...

ပထမဦးဆုံး Spot Instance ဆိုတာ AWS ရဲ့ Spare compute capacity လိုဆိုနိုင်ပါတယ် AWS ရဲ့ ပိုနေတဲ့ compute resource တွေကို လေလံပုံစံနဲ့ ပြန်လည် ငှားရမ်းတဲ့သဘောပါ ဆိုတော့ လေလံပုံစံဆိုတော့ ကိုယ်ပေးတဲ့စျေးက များနေရင် ရမယ် နည်းသွားမယ်ဆို မရတော့ဘူးပေါ့၊ ဒါဆို ဘာဖြစ်မလဲ ပြန်သိမ်းသွားပီး တခြားပိုပေးတဲ့ လူဆီ ကို ငှားရမ်းလိုက်မှာပါ နောက်ပီး EC2 Demand များလာတဲ့အခါပြန်ယူသွားမယ် ဒါပေမဲ့ မစိုးရိမ်ပါနဲ့ ကိုယ်က တခြားကျန်နေတဲ့ resource တွေကို ရယူလို့ရပါသေးတယ်၊ ရရှိခဲ့ရင်လည်း အရမ်းကိုစျေးသက်သာတဲ့ စျေးနဲ့ရမှာ ပါ .Official Documentation မှာ ပြောထားတာ က On-Demand စျေးထက် 90% လောက်ထိသက်သာတယ် ပြောထားပါတယ်။ ဒါဆို Spot instance တွေကို ဘာကြောင့် ဘယ်လိုနေရာမျိုးတွေ မှာ အသုံးပြုရမှာ လဲ မေးစရာရှိလာပါတယ်၊

AWS Spot Instance တွေကို အထူးသဖြင့် အရေးမကြီးတဲ့ Workloads, Compute Capacity များများလိုပီး ခဏပဲ run မည့် Usecase ဥပမာ Bigdata Batch Process တွေ, Stateless Application တွေကို Auto Scale လုပ်တဲ့နေရာ, CI/CD workload နဲ့ အခြားသော Container Cluster Nodes provision workload တွေမှာ အသုံးပြုနိုင်ပါတယ်။ အထူးသဖြင့် ပျက်သွားလည်းမလွမ်းလောက်ပါဘူးဆိုတဲ့ နေရာတွေမှာအသုံးပြုနိုင်ပါတယ်။ နောက်ပီး Microservices ရဲ့ Cattle analogy အတိုင်း Dynamic infrastructure တစ်လုံးမရှိတစ်လုံး ရော့ သွားစမ်း နောက်တစ်လုံးလာဆိုပီး စျေးသက်သက်သာသာနဲ့ ဖောဖောသီသီသုံးချင်တဲ့ သူများ အတွက်ဆို Spot Instance က ကိုက်ညီပါတယ်။



ပုံမှာမြင်ရတဲ့ အတိုင်းဆိုရင် EC2 Instance T3.medium ရဲ့ On-demand price က $0.0528/hr ကျသင့်မှာဖြစ်ပြီးတော့ Spot Price နဲ့ဆိုရင်တော့ $0.0158/hr ပဲကျသင့်မှာဖြစ်ပါတယ်။ 70% လောက်ကိုသက်သာပါတယ်။ Instance Size ကိုမှုတည်ပြီး ဈေးနှုန်းတွေလည်း မတူနိုင်ပါဘူး။ တစ်ချို့ Instance Type တွေဆိုရင် 70%-90% လောက်ထိသက်သာပါတယ်။



ဒီပုံမှာပြထားတာကတော့ Saving Summary ပါ။ ကျွန်တော်သုံးထားတာကတော့ T3.medium instance ၂လုံးကို ၃ရက်လောက်သုံးထားပါတယ်။ On-demand Price ဆိုရင်တော့ $6.49

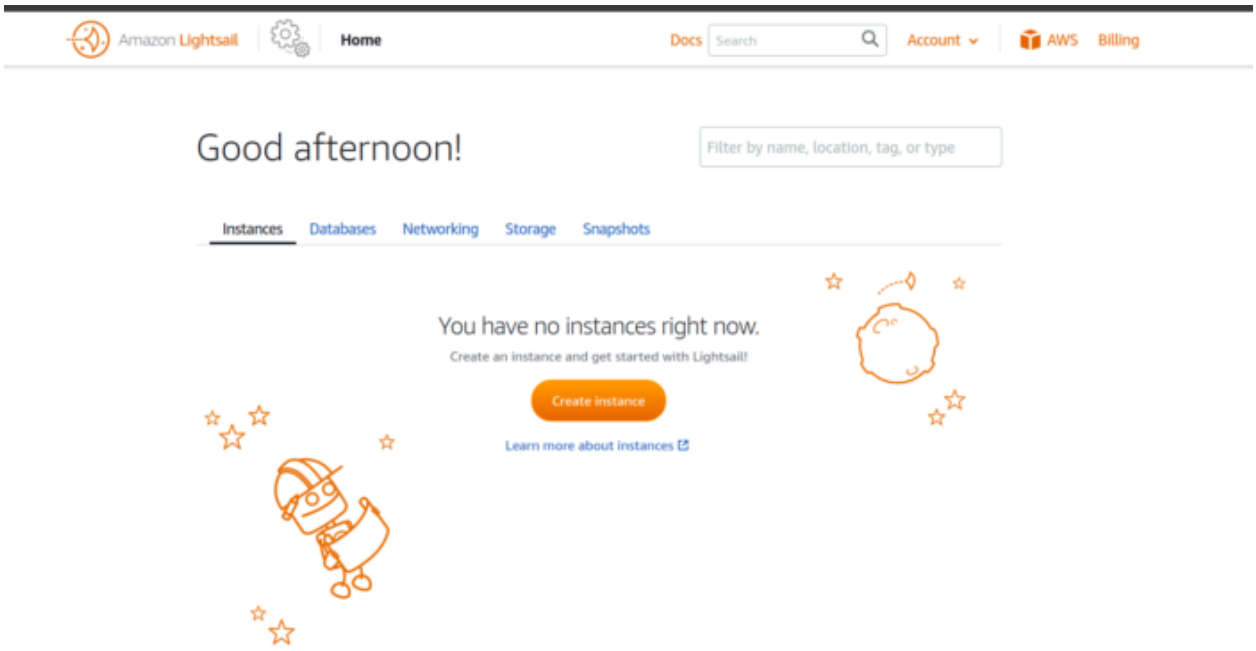
လောက်ကျသွားမယ်ပေါ့နော်။ Spot Instance နဲ့သုံးမယ်ဆိုရင်တော့ ၃ရက်စာ ၇၂နာရီကို $1.94 လောက်ပဲကျပါတယ်။ ဒါ့အပြင် ဈေးအရမ်းသက်သာနေလို့ On-demand Instance တွေလို Performance ကောင်းရဲ့လား၊ Specification ရောလျော့သွားလား ဘာညာစတာတွေ ပူစရာမလိုပါဘူး၊ Spot Instance ကလည်း On-Demand Instance တွေကိုသုံးသလိုပဲ အလွယ်တစ်ကူအသုံးပြုလို့ရပါတယ်။ ပြောင်းလဲလာတဲ့ Spot Instance Pricing Model အရလည်း အရင်လို Bidding တွေအတွက် ခေါင်းရှုပ်စရာမလိုတော့ပါဘူး။

ဒီလောက်ဆိုရင်တော့ EC2 Spot Instance အကြောင်းကို အနည်းနဲ့အများနားလည်သဘောပေါက်လောက်ပြီထင်ပါတယ်။ နောက်ပိုင်း Article တွေမှာလည်း Spot Instance နဲ့ပတ်သက်တဲ့ Spot Block, Spot Fleet နဲ့ Best Practices တွေကိုရေးသားဦးမှာဖြစ်လို့ ဆက်လက်အားပေးကြပါဦးခင်ဗျာ။

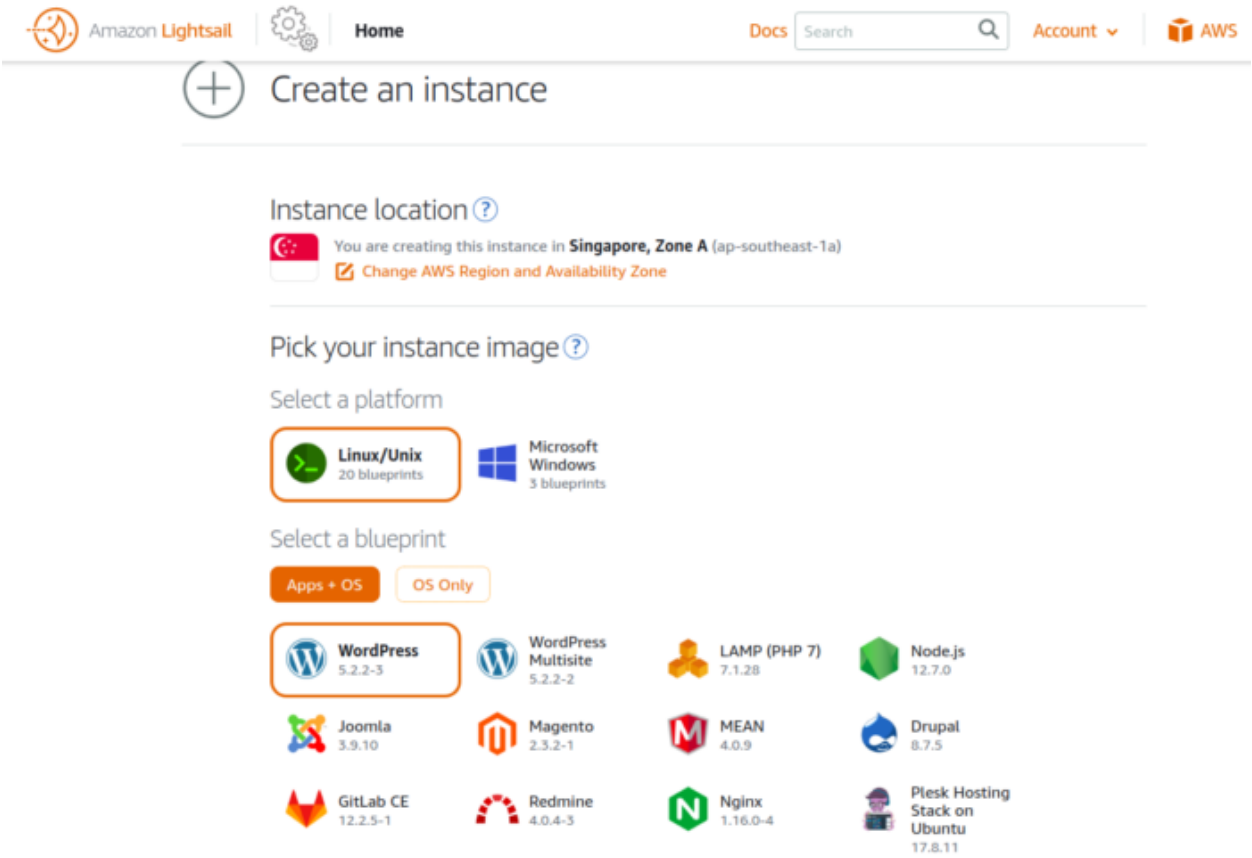
Deploying Wordpress Website with Amazon Lightsail

အခုပြောပြပေးချင်တာကတော့ WordPress ဆိုဒ်တခုကို AWS မှာ ဈေးသက်သက်သာသာနဲ့ Host ဘယ်လိုလုပ်မလဲဆိုတာမျိုးပဲ ဖြစ်ပါတယ်။ Server Configuration တွေသိစရာမလိုပဲ အလွယ်တကူ Build လုပ်လို့ရတဲ့ services တမျိုးပါ။ နောက်ကျရင် Pros & Cons လည်း သီးသန့်ရေးပေးပါအုံးမယ် အချိန်ရတာနဲ့အမျှ အခုတော့ setup လုပ်တာလေးကိုပဲ အဆင်ပြေရင်လိုက်စမ်းကြည့်ပီး သုံးကြည့်ကြပါလို့ပြောချင်ပါတယ်။

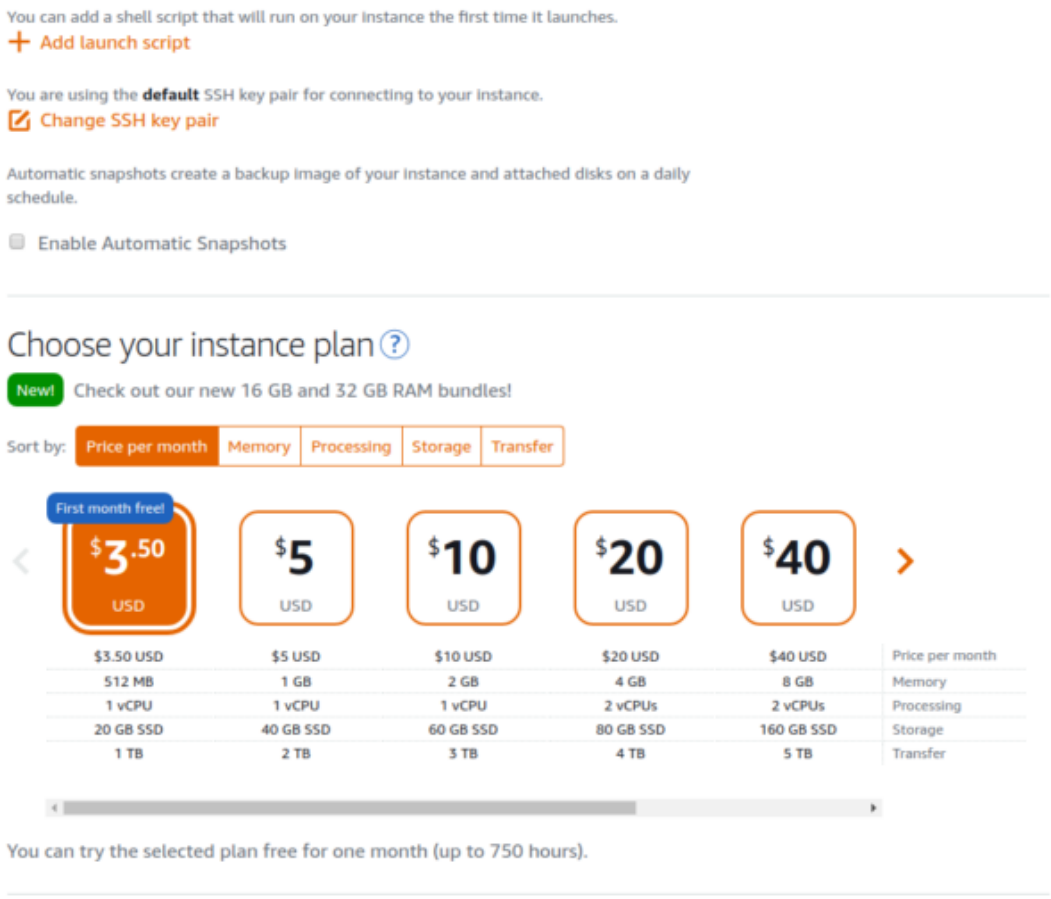
ပထမဆုံးအနေနဲ့ AWS Lightsail (https://lightsail.aws.amazon.com/ls/webapp/home/instances) ရဲ့ Console ကိုအရင် ဝင်လိုက်ပါ ပုံမှာပြထားတဲ့အတိုင်းမြင်ရပါလိမ့်မယ်



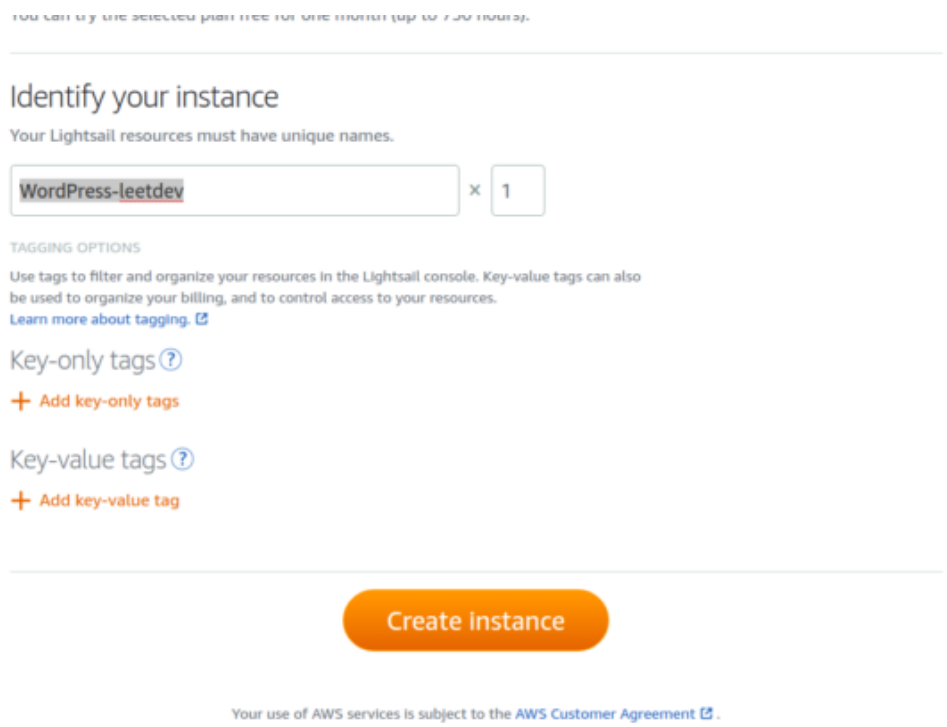
Create Instance ကနေ မိမိ အသုံးပြုလိုသော Application အမျိုးအစားတွေအလိုက် ရွေးလို့ရပါတယ်။ ကျွန်တော်ကတော့ WordPress ကိုအဓိက ပြောမှာ ဖြစ်လို့ WordPress ကိုပဲ ရွေးလိုက်ပါတယ်။ WordPress site အများကြီးလုပ်ချင်ရင်လည်း ရပါတယ် ဘေးက တခုကို ရွေးပေးပီး လုပ်လို့ရပါတယ်။



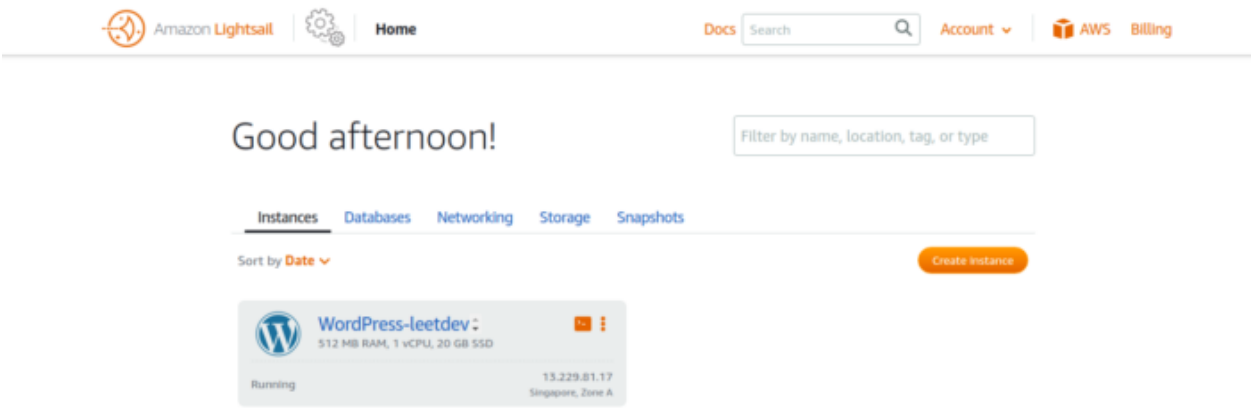
နောက်တဆင့်ကတော့ ကိုယ်သုံးမယ့် WordPress site အတွက်လိုအပ်မယ် Specification လေးတွေ ရွေးပေးလို့ရပါတယ် ကိုယ့်ဆိုဒ်ရဲ့ User ဘယ်လောက်ထိရှိမယ် Request များလား Response များလားပေါ်မူတည်ပီး ကိုယ်စိတ်ကြိုက် memory(RAM) တွေရော storage ရော ရွေးပေးလို့ရပါတယ်။ အောက်က ပုံမှာ မြင်နိုင်ပါတယ်။ default ssh key pair မသုံးပဲ မိမိစိတ်ကြိုက်လည်း ပြောင်းရွေးပေးခဲ့လို့ရပါတယ်။



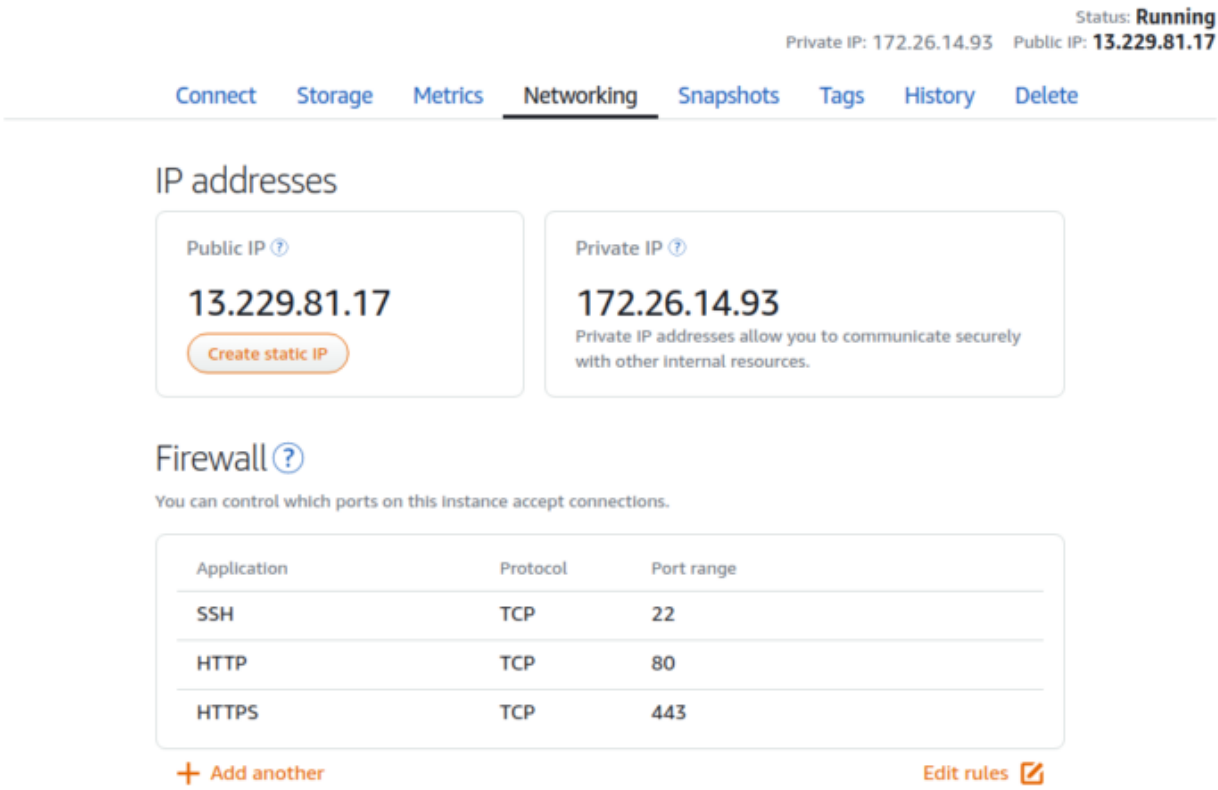
ကိုယ်ပေးခြင်တဲ့ နာမည်နဲ့ Create Instance ရသွားပီးဆိုရင် WordPress site လေးတော့ ပီးသွားပီလို့ ယူစလို့ရပါတယ်။



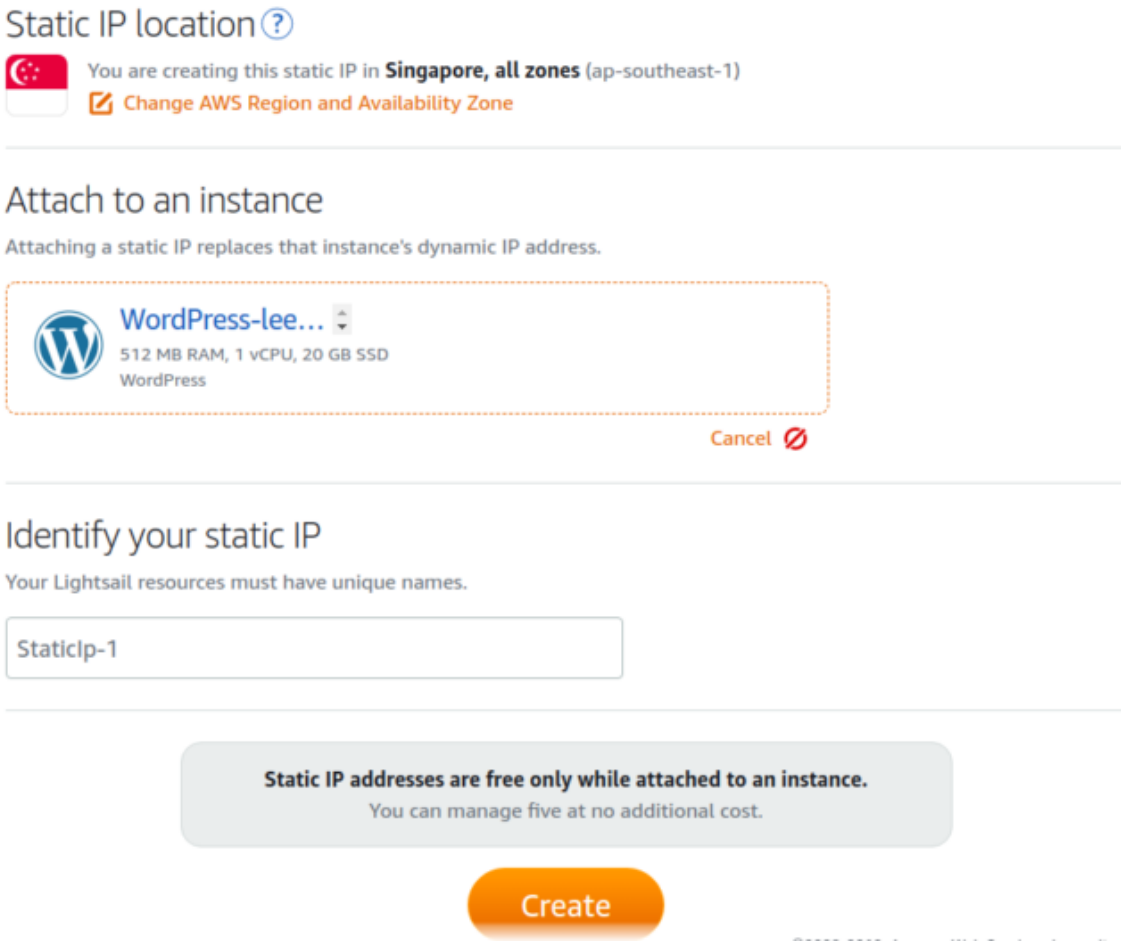
Running ဖြစ်နေတဲ့ WordPress လေးကို ခဗျားတို့ တွေ့ရမှာ ဖြစ်ပါတယ်



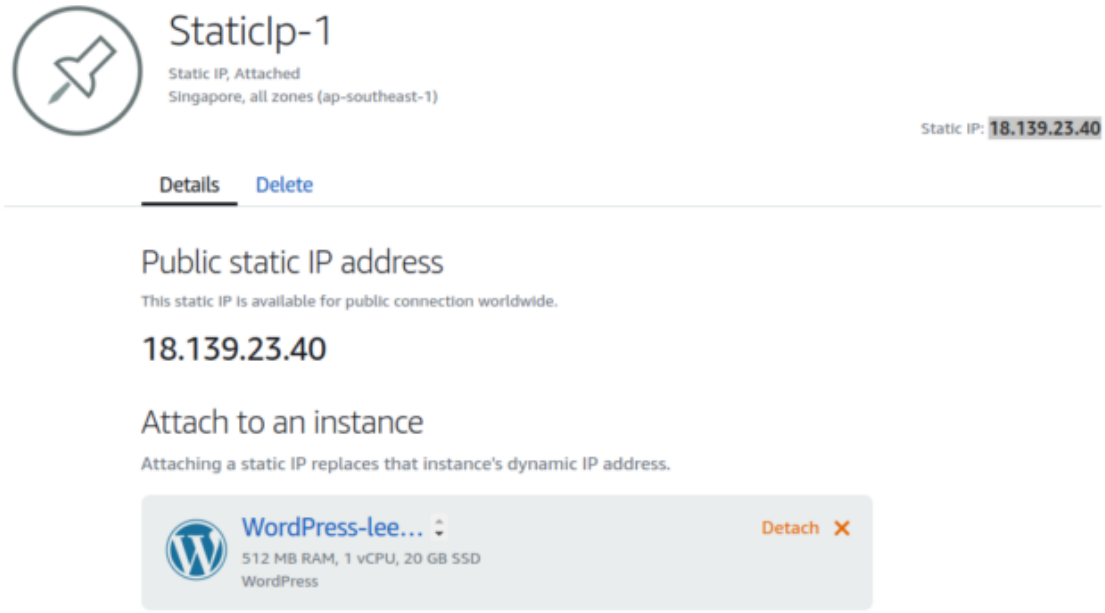
နောက်တခုကတော့ ကျွန်တော်တို့ အနေနဲ့ Create လုပ်ထားတဲ့ WordPress ကို public ip နဲ့ခေါ်လို့ရအောင် elastic ip (static ip) တခု Attach လုပ်ပေးဖို့လိုပါတယ်။ Networking ဆိုတဲ့ tag ကနေသွားလုပ်ရမှာပါ ပုံမှာပါတဲ့အတိုင်း Create Static IP လုပ်ရမှာပါ။



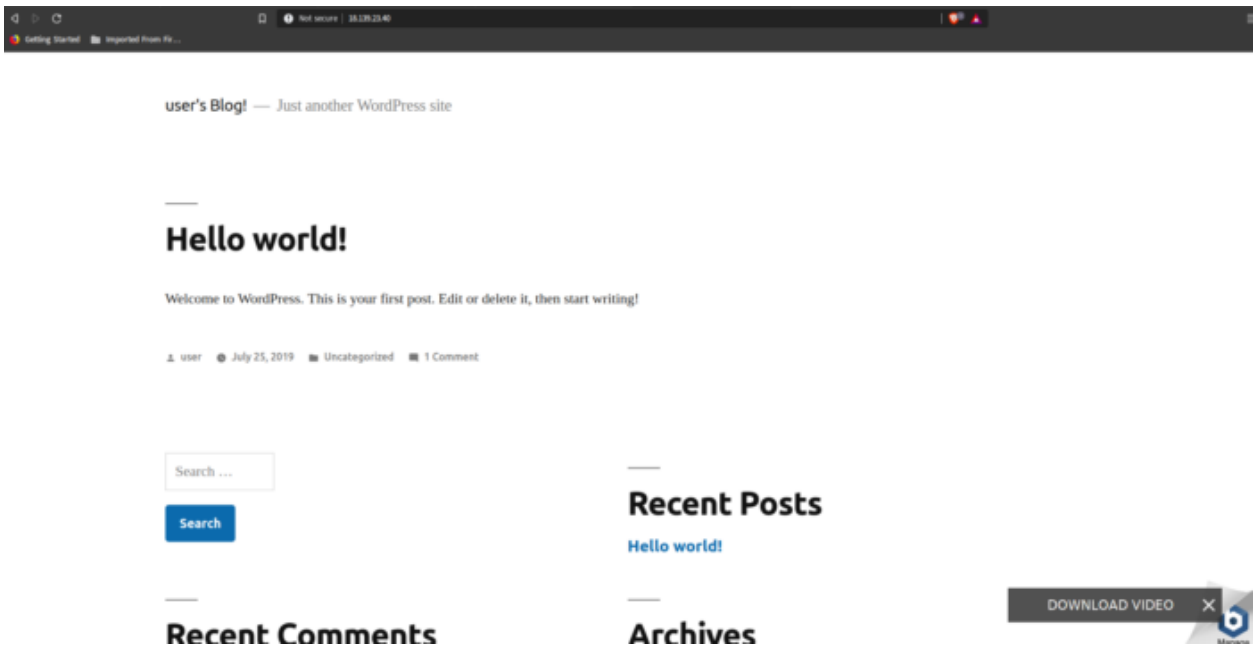
Create Static IP လုပ်တဲ့အခါမှာ ဘယ်အလုံးကို Attach လုပ်မလဲရွေးပေးရပါတယ် ကျွန်တော်တို့ အနေနဲ့ WordPress Leetdev ကို ရွေးပေးလိုက်ပါပြီ Create လိုက်ရင် ရပါပြီ။ သူအစကတည်းက ပေးထားတဲ့ Public IP သုံးလို့လည်းရပါတယ်။



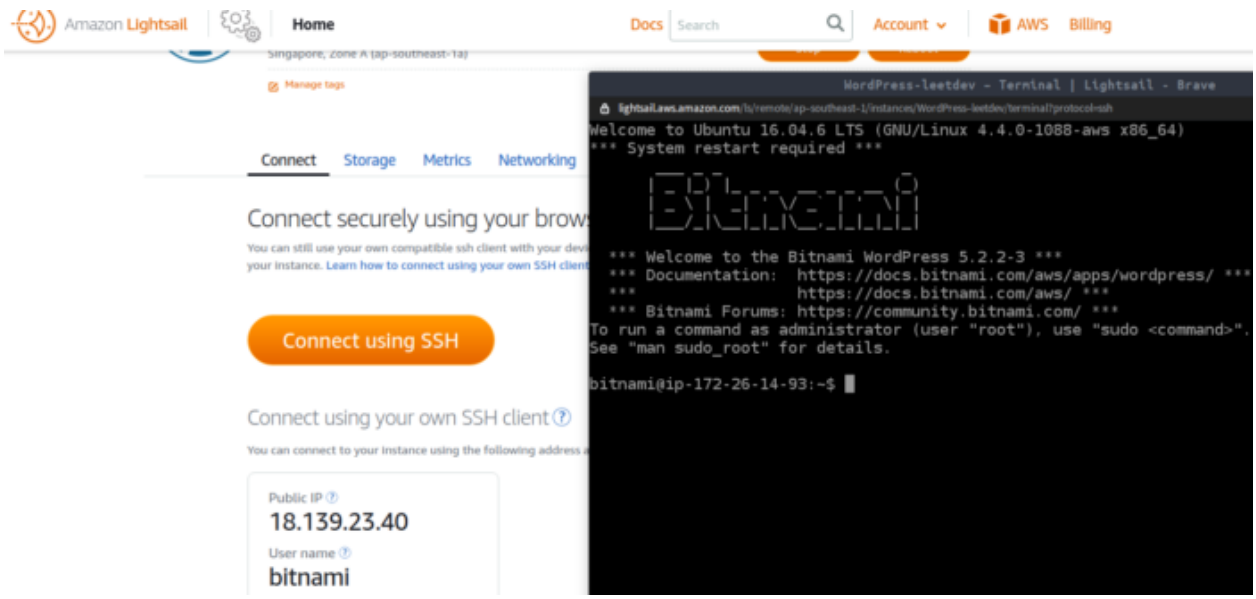
အခုဆိုရင် ကျွန်တော်တို့ wordpress ဆိုဒ်ကို ခေါ်လို့ရမယ့် Public IP တခုရလာပါပြီ AWS Route53 နဲ့ Domain ဘယ်လို Point မလဲဆိုတာကိုတော့ ကျွန်တော်နောက် အချိန်ရရင် ရေးပေးပါအုံးမယ်။



အောက်ကပုံကတော့ WordPress setup လုပ်ထားပီးသား Default ဆိုဒ်တခုရလာပါပြီ ကျွန်တော်တို့အနေနဲ့ မိမိတို့သုံးလိုတဲ့ Themes တွေ Plugin တွေတော့ ထည့်ပီး လုပ်ဖို့တော့လိုပါအုံးမယ်။



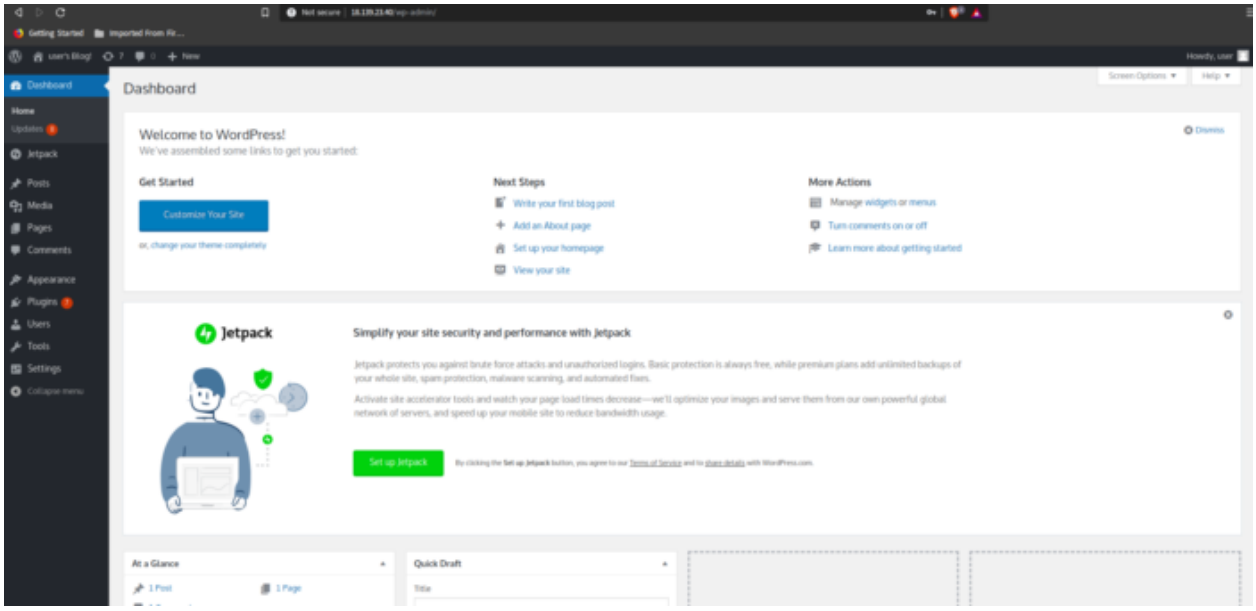
နောက်တခုကတော့ wp-login ကနေ ဝင်ဖို့ ကျွန်တော်တို့ အနေနဲ့ username နဲ့ password ရဖို့လိုပါသေးတယ်။ ဘယ်ကနေရမလဲဆိုတော့ ကျွန်တော်တို့ အနေနဲ့ WordPress Lighsail server ထဲကို ssh login ဝင်ရမှာပါ ကျွန်တော်ကတော့ သူ့ Console ကနေပဲ Login ဝင်လိုက်ပါတယ်။



ls ခေါ်ကြည့်လိုက်ပါ credentials တွေပေးထားတာ တွေ့ရပါလိမ့်မယ်။ cat နဲ့ ဖတ်ကြည့်လိုက်တဲ့အခါ wp-login အတွက် username & password တွေ့ရပါလိမ့်မယ်။

```
bitnami@ip-172-26-14-93:~$ cat bitnami_credentials
Welcome to the Bitnami WordPress Stack

******************************************************************
The default username and password is 'user' and 'jDtllYvhZc
******************************************************************

You can also use this password to access the databases and
ncludes.

Please refer to https://docs.bitnami.com/ for more details.

bitnami@ip-172-26-14-93:~$
```

ကဲရလာတဲ့ Credential တွေကိုသုံးပြီး wordpress site ကို login ဝင်လို့ရပါပြီ မိမိစိတ်ကြိုက် wordpress site တခုကို အမြန်ဆုံး develop လုပ်နိုင်သွားပီ ဖြစ်ပါတယ်။



အားလုံးပဲ အဆင်ပြေကြမယ်လို့ထင်ပါတယ် ဖတ်ပေးကြသူအားလုံးကို ကျေးဇူးတင်ပါတယ်။

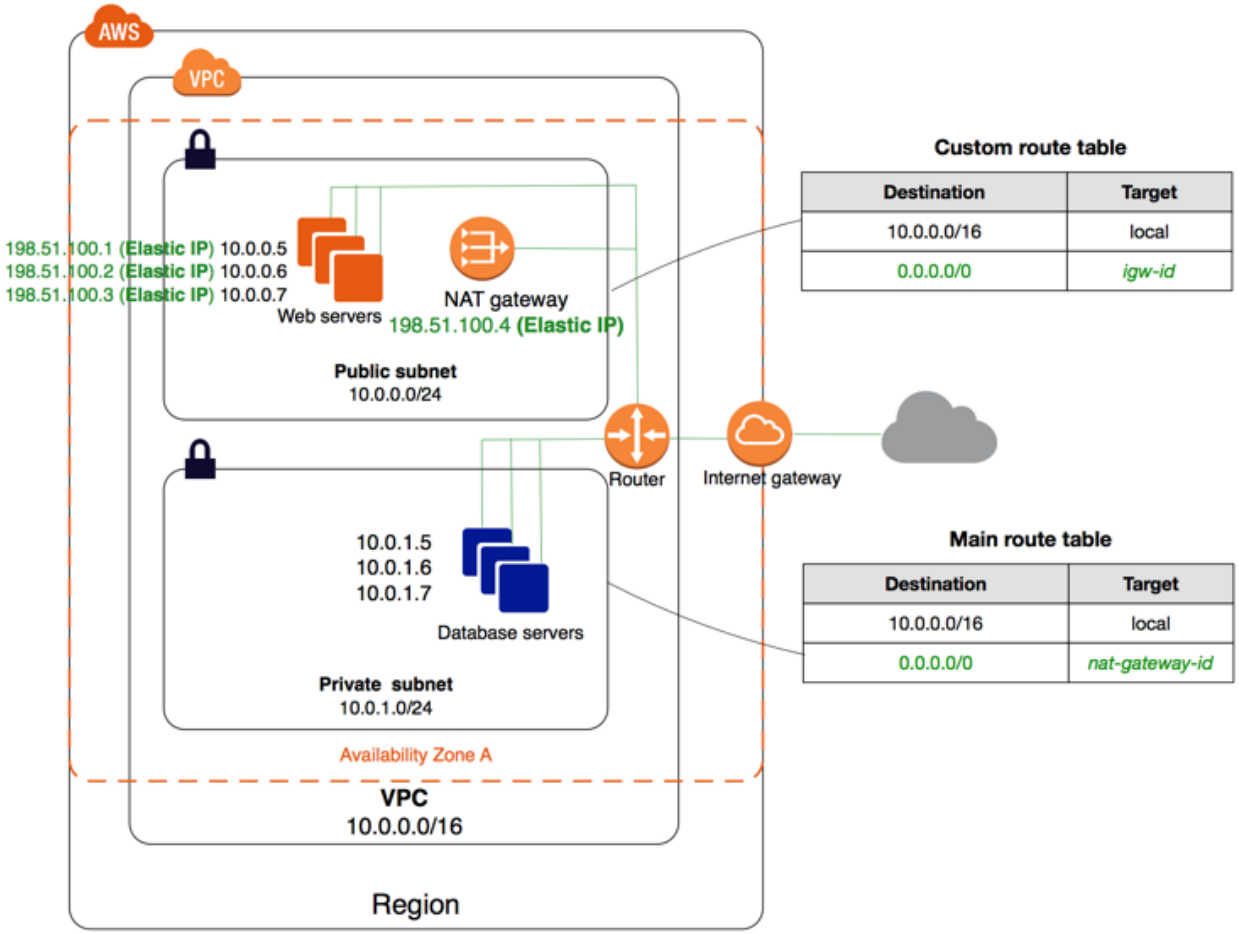
Feel the power of Auto Scaling and Spot Instances with auto assign EIP

Auto Scaling and Spot Instance with auto assign EIP use case

ဒီသင်ခန်းပိတ်ရက်မှာ အပြင်လည်းမထွက်ရတော့ ပျင်းပျင်းနဲ့ ဂိမ်းဆာဗာလေးတစ်ခုလုပ်ဖို့ စဉ်းစားမိရင်ကနေ ကိုယ့်ပိုက်ဆံနဲ့ကိုယ်လုပ်မှာဆိုတော့ တွက်ချေလည်းကိုက်ရမယ် Server ကိုလည်း **Uptime 99.5%** လောက်ပေးချင်တယ်ဆိုပြီး Infra ပိုင်းလေးနည်းနည်းစဉ်းစားကြည့်ပြီး လုပ်ဖြစ်တာလေးကို ဝေမျှလိုက်ရပါတယ်။

ဒီနေရာမှာ ကျွန်တော့်အနေနဲ့ က လိုအပ်တာတွေရှိတယ်။ အဲ့ဒါတွေကို အရင်စဉ်းစားလိုက်တယ်။ ဘာလိုမလဲဆိုရင် ထိုးစံအတိုင်း အရင်ဆုံး Infrastructure requirement အရ Virtual Private Cloud (VPC) တစ်ခုကို Public / Private Subnets တွေနဲ့ တည်ဆောက်လိုက်တယ်။ Public / Private Subnets ဆောက်တာက ဘာအတွက်လဲဆိုတော့ အရင်ဆုံးဘာမှမရှိသေးတဲ့ Infrastructure မှာ Network လိုလို့ပါ။ Network Diagram အရဆိုရင်တော့ အောက်မှာပြထားတဲ့ပုံထဲကလိုမျိုးပေါ့ဗျာ ။ ပြီးတော့ Web Server နဲ့ Application Server အပြင် ကျွန်တော်သုံးချင်တဲ့ထဲမှာ Database Server တစ်ခုသုံးဖို့လိုလို့။ Best Practices ဆိုတာထပ်လည်း Database Server လိုမျိုးက Public Subnet မှာမထားသင့်ထဲမှာပါတယ်။ Best Practices အရလည်း အဆင်ပြေတယ်။ အဆင်မပြေတာက တစ်ခုပဲ။ ဘာလဲဆိုတော့ NAT Gateway အတွက်ပိုက်ဆံထပ်ကုန်တာပါ။ အဲ့နေရာမှာ Private Subnet က Internet ကို ထွက်ဖို့အတွက် **NAT Gateway** လိုတယ်။ ဒီနေရာမှာ ဈေးသက်သာချင်ရင် Web , App, Database သုံးခု စလုံးကို

Public Subnet မှာထားပြီးလုပ်လိုက်ရင်လည်း အဆင်ပြေတယ်။ အဲ့ကြရင် NAT Gateway ဖိုးမကုန်တော့ဘူး။ ဘယ်လိုမှုဖြစ်မယ်ဆိုတာမျိုးတော့ မရှိပါဘူး ။



Network Diagram with NAT Gateway

ကျွန်တော်ကတော့ Web နဲ့ Application Server ကို Public Subnet မှာထားပြီးတော့ Database ကိုတော့ Private Subnet ထဲပို့လိုက်တယ်။ VPC Setup ပြီးပြီဆိုရင်တော့ ကျွန်တော်တို့လိုချင်တဲ့ Game Server လုပ်ဖို့ Compute Instances (EC2) ဘက်အပိုင်းကိုသွားလိုက်ရအောင်။ ကိုယ်သုံးချင်တဲ့ Tier ပုံစံကို တစ်ခုထပ်စဉ်းစားဖို့လိုပါမယ်။ ဘယ်ဟာကို ဘယ်လိုထားမှာလဲ 2 Tier လား 3 Tier လား ၊ ဘယ်နေရာမှာ ဘာသုံးမယ် စသည်ဖြင့်ပေါ့ဗျာ။ ကျွန်တော်ကတော့ Web, App, Database သုံးခုမှာ Web နဲ့ App ကို Instance တစ်ခုတည်းသုံးလိုက်တယ်။ ပိုက်ဆံကုန်သက်သာအောင်လို့ပါ ;) Database Server ကိုတော့ AWS ရဲ့ Managed RDS ကိုမသုံးတော့ဘူး ဒီနေရာမှာ ကိုယ်တိုင် EC2 ကို Setup လုပ်ပြီး Manage လုပ်ပါမယ်။ အဲတော့ စုစုပေါင်းမှ EC2 Instance ၂ ခုပဲလိုပါတယ်။

အိုကေ အဲ့တော့ Application Server အတွက် အမြဲတမ်း Run နေအောင် နဲ့ အကုန်အကျလေးတော်တော်သက်သာအောင် EC2 AutoScaling နဲ့ Spot Instance ကိုတွဲသုံးပါမယ်။ အပေါ်မှာပြောခဲ့သလို ပဲ Uptime က 99.5% ရှိနေရမှာဖြစ်တဲ့အတွက် Server ကတစ်ခုခုဖြစ်သွားရင် ချက်ချင်းဆိုသလိုပြန်တက်လာပြီး Up and Running ဖြစ်နိုင်ဖို့ လိုပါတယ်။ အဲ့အတွက် AWS ရဲ့ EC2

Auto Scaling သုံးရတာပါ။ ဈေးသက်ဖို့အတွက် နောက်တစ်ခုက **EC2 Spot Instance** ပါ။ Spot Instance အကြောင်းကို အောက်က လင့် မှာလည်းလေ့လာလို့ရပါတယ်။

Reduce cloud compute cost with EC2 Spot Instance
/compute/reduce_cloud_compute_cost_with_ec2_spot_instance

သူကတော့ EC2 Instance ရဲ့ Purchase Option တစ်ခုပါ ပုံမှန် Instance (On-demand) ထပ်များသောအားဖြင့် Average ဈေးက 60-70% လောက်သက်သာပါတယ်။
နောက်တစ်ခုလိုအပ်တာကတော့ **Elastic IP** ပါ။ သူကတော့ Auto Scaling မှာ EC2 Instance က တစ်ခုခုဖြစ်ပြီး Down သွားရင်သော်လည်းကောင်း Spot Instance နဲ့တွဲသုံးမှာဖြစ်တဲ့အတွက် Spot Instance ရဲ့ သဘောတရားအရ Spare Resource တွေမလောက်တော့ရင်ဖြစ်ဖြစ် ကိုယ့် EC2 Instance ကပျက်သွားမှာပါ။ အဲ့ဒီအခါကြရင် Auto Scaling နဲ့ EC2 Instance အသစ်တစ်ခုပြန်တက်လာတဲ့ အခါမှာ အရင်ကရှိခဲ့တဲ့ EC2 Instance ရဲ့ IPv4 IP Address ကပျက်သွားပြီးတော့ IP Address အသစ်တစ်ခုနဲ့ပြန်တက်လာမှာပါ။

ဒီမှာ ပြဿနာက အရင် EC2 Instance ရဲ့ IP Address ကသာ Domain နဲ့တွဲထားမယ်ဆိုရင် Instance အသစ်ပြန်တက်လာတဲ့အခါမှာ IP ပြောင်းသွားတဲ့အတွက် Access လုပ်တဲ့အခါမှာ အခက်တွေ့စေမှာဖြစ်ပါတယ်။

Elastic IP သုံးထားတဲ့အတွက် Auto Scaling Group မှာ EC2 Instance က အပေါ်မှာပြောခဲ့တဲ့ Case တစ်ခုခုကြောင့်က Down သွားတယ်ဆိုရင် အသစ်တက်လာမဲ့ EC2 Instance မှာ Elastic IP က auto assign လုပ်သွားမှာဖြစ်တဲ့အတွက် စိတ်ပူစရာမလိုတော့ပါဘူး။ ထပ်ဆောင်းပြောရရင် ကျွန်တော်တို့ အခု use case မှာဆိုရင် Spot Instance သုံးထားတဲ့အတွက် ပိုတောင်လိုအပ်ပါသေးတယ်။ အခုဆိုရင် လိုအပ်တွေရပြီဆိုတော့ Web/App Server အတွက် EC2 Spot Instance ကို Auto Scaling အသုံးပြုပြီး တည်ဆောက်ပါမယ်။

Auto Scaling Group သုံးပြီး EC2 Instance တည်ဆောက်တဲ့အခါမှာ ရေရှည်အတွက် အဆင်ပြေအောင် **Launch Template** လေးတစ်ခုပါ create လုပ်လိုက်ပါတယ်။ Launch Template ဆိုတာကတော့ အလွယ်ပြောရရင် VM Template ပေါ့ဗျာ။ ဘာတွေပါမယ် ဘယ်လိုလုပ်မယ်ဆိုတာကြိုတင် Define လုပ်ထားလို့ရတယ်။ သူကတော့ Auto Scaling နဲ့တွဲသုံးဖို့ပါ။ Auto Scaling Group က Instance တွေအတွက် ဘယ် Image (AMI) ကိုသုံးမယ် Security Group (SG) ကဘာထားမယ် ဘယ် Instance size သုံးမယ် ဆိုတာမျိုးတွေကိုပေါ့။ ဒီအတိုင်း EC2 Instance ကို Create လုပ်ပြီး စိတ်ကြိုက်လိုတာတွေသွင်းပြီမှ Launch Template ကိုပြန်ပြီး create လုပ်လို့လည်းရပါတယ်။ Auto Scaling Group အတွက် Launch Template တစ်ခုဘယ်လို create လုပ်လဲဆိုတာကိုတော့ အောက်ကလင့်မှာလေ့လာနိုင်ပါတယ် ။

https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-launch-template.html

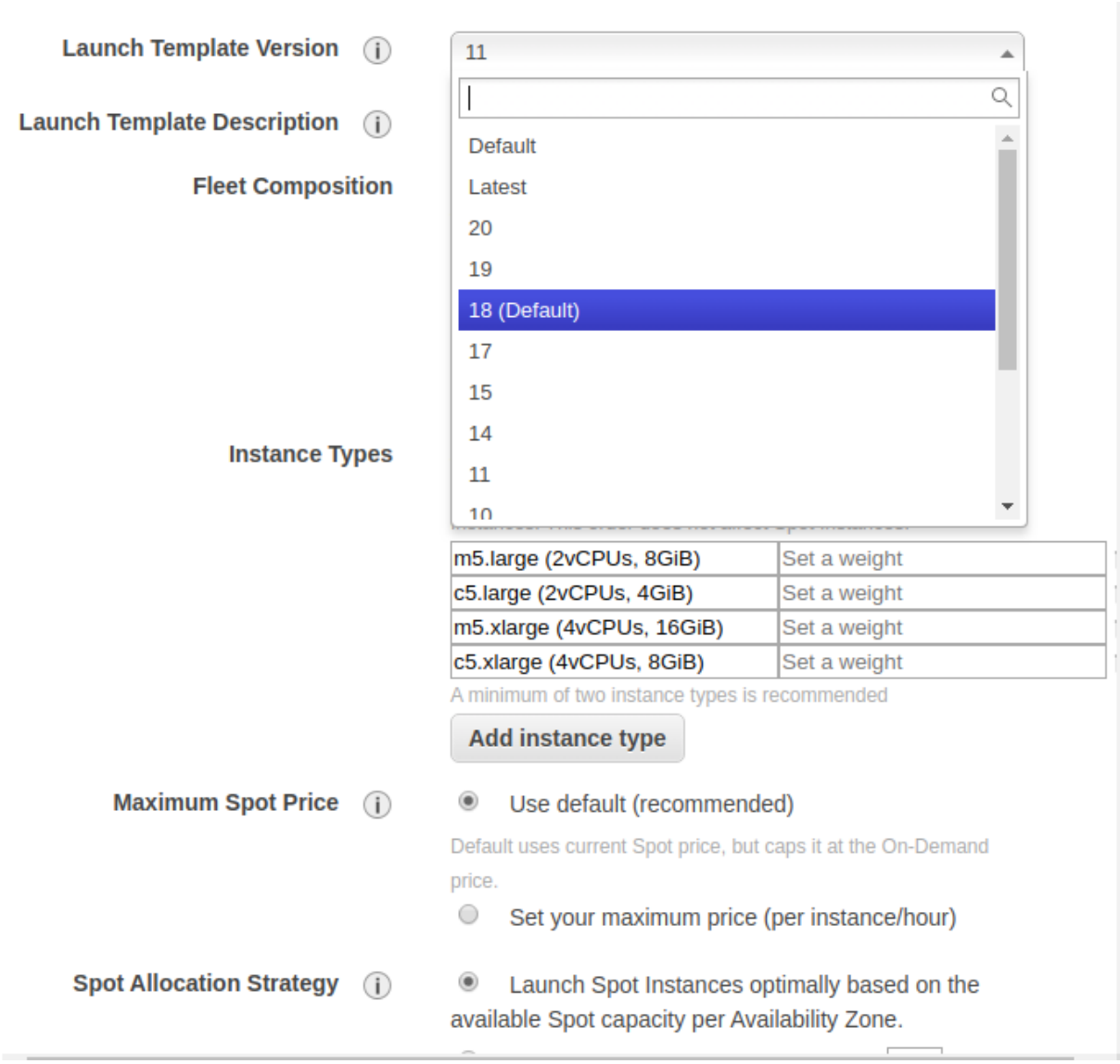
ဒီမှာနောက်ထပ်တစ်ခုရှိတာက Auto Scaling Group ကို Launch Template နဲ့မဟုတ်ပဲ **Launch Configuration** ဆိုတာနဲ့လည်းတွဲသုံးလို့ရတယ်ဆိုတာပါပဲ။ ဒါပေမယ့် Launch Template ကပိုလုပ်လို့ရပါတယ်။ ဆိုလိုတာက Launch Template မှာ များများပိုကလိလို့ရတယ်။ အဓိကအားသာတာ တစ်ခုကတော့ Launch Template က Versioning လုပ်လို့ရတာပဲ။

Launch Template Sample

Versioning လုပ်လို့ရတဲ့ အတွက် ကျွန်တော်တို့ အခု use case
အတွက်လည်းတော်တော်အဆင်ပြေပါတယ်။

Launch Template ကို create လုပ်ပြီးရင်တော့ Auto Scaling Group ကို Spot Instance နဲ့ စတင် create
လုပ်လို့ရပါပြီ။ Launch Template ထဲမှာတော့ ကျွန်တော်တို့က EC2 Instance ကိုဘယ် Subnet
မှာထားမယ်ဆိုတာတွေကို Define မလုပ်ခဲ့တဲ့အတွက် Auto Scaling Group မှာ Define လုပ်ရပါမယ်။

/App Server နဲ့ Database Server ကို Same Subnet မှာထားရင် Latency
မဆိုသလောက်နည်းနည်းလေး ရှိတဲ့ issue ကိုအဆင်ပြေနိုင်ပါတယ်။ Same Subnet ဆိုတာထပ် Same
Availability Zone (AZ) တူတူထားတာကို ပြောတာပါ။ AZ မတူရင်ဘာဖြစ်လဲဆိုတော့ Millisecond
latency မှာသိသာပါတယ်။



auto scaling setting 1

နောက်တစ်ခုက Launch Template မှာ EC2 Instance Type တွေကို Define လုပ်ခဲ့ပေမယ့် Auto Scaling ရောက်မှလည်း Define လုပ်လို့လည်းရသလို Override လုပ်လို့လည်းရပါတယ်။ နောက်ထပ် Instance Type တွေလည်းထပ်ထည့်လို့ရပါတယ်။ Option 1 ကတော့ Launch Template မှာ Define လုပ်ခဲ့တဲ့ Instance Type အတိုင်းသုံးမှာပါ။ Option 2 ကတော့ စိတ်ကြိုက်ဖန်တီးလို့ရတဲ့ Option ပေါ့။ နောက်ထည့် Instance အသစ်တွေပြောင်းတာ တွေဘာတွေ အတွက်အသုံးတည့်ပါတယ်။ ကျွန်တော်ကတော့ Option 2 နဲ့ရွေးလိုက်ပါတယ်။

Option 1

Fleet Composition

Adhere to the launch template

The launch template determines the instance type and purchase option (On-Demand or Spot).

Combine purchase options and instances

Choose a mix of On-Demand Instances and Spot Instances and multiple instance types. Spot Instances are automatically launched at the lowest price available.

Desired Capacity 1

Min 1

Max 3

Availability Zone(s) ap-southeast-1b ×

Option 1

Option 2

Fleet Composition

○ Adhere to the launch template

The launch template determines the instance type and purchase option (On-Demand or Spot).

◉ Combine purchase options and instances

Choose a mix of On-Demand Instances and Spot Instances and multiple instance types. Spot Instances are automatically launched at the lowest price available.

Instance Types

Add acceptable instance types and optional weights to your fleet. Change their order to set the launch priority of On-Demand Instances. This order does not affect Spot Instances.

c5.xlarge (4vCPUs, 8GiB)	Set a weight
m5.xlarge (4vCPUs, 16GiB)	Set a weight
c5.2xlarge (8vCPUs, 16GiB)	Set a weight

A minimum of two instance types is recommended

Add instance type

Maximum Spot Price ⓘ

◉ Use default (recommended)

Default uses current Spot price, but caps it at the On-Demand price.

○ Set your maximum price (per instance/hour)

Spot Allocation Strategy ⓘ

◉ Launch Spot Instances optimally based on the available Spot capacity per Availability Zone.

○ Diversify Spot Instances across your [2] lowest priced instance types per Availability Zone

Optional On-Demand Base ⓘ

Designate the first [0] instances as On-Demand

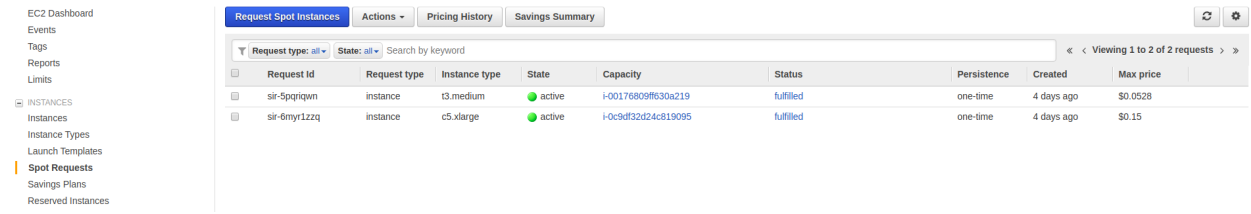
On-Demand Percentage ⓘ

[0] % On-Demand and 100% Spot

Option 2

Auto Scaling Group ကို Create လုပ်ပြီးပြီဆိုရင်တော့ ကိုယ့်ရဲ့ Desire Capacity အလိုက် EC2 Instance တွေကို create လုပ်သွားပါလိမ့်မယ်။

ကျွန်တော်တို့က Auto Scaling ကို Spot Instance သုံးထားတဲ့ အတွက် EC2 Instance က Spot Instance ကိုသူ့ဘာသူတည်ဆောက်ပေးသွားမှာပါ။ AWS Console ထဲက Spot Request ဆိုတဲ့ Tab မှာသွားကြည့်လို့ရပါတယ်။



Spot Request

ဒါဆိုရင်တော့ Auto Scaling Group အကူအညီနဲ့ Spot Instance ကို တည်ဆောက်လို့ပြီးပါပြီ ။ ဒီထိအပြီးမှာ ဘာဖြစ်မလဲဆိုတော့ အပေါ်မှာပြောခဲ့သလို ASG သုံးထားတာဖြစ်တဲ့ အတွက် Instance တစ်လုံး Down သွားရင် အသစ်တစ်လုံး Auto ပြန်တက်လာမယ်၊ Spot Instance ကို သုံးပါဆိုပြီး ရွေးထားတဲ့ အတွက် On-Demand Instance ထပ်ဈေးကပိုသက်သာမယ်။ ဒါဆိုကိုယ်လိုချင်တာနဲ့ နည်းနည်းကိုက်လာပြီ။ ဒီမှာတင်ပြီးသွားမှာလားဆိုတော့ မဟုတ်သေးဘူး။ အသစ်တက်လာတဲ့ Instance ကို Elastic IP auto associate လုပ်ချင်တာကျန်သေးတယ်။ EIP auto associate လုပ်ဖို့က EC2 Instance မှာ Permission လိုအပ်တယ်။ အဲ့တော့ ကျွန်တော်တို့က EC2 Instance ကနေ သူ့ဘာသာသူ Elastic IP ကို auto associate လုပ်နိုင်ဖို့ **IAM Role** တစ်ခုဆောက်ပေးရပါမယ်။ IAM Role မှာက Permission polices တွေ attach လုပ်ဖို့အတွက် **IAM Policies** ကိုအရင် create လုပ်ရပါမယ်။

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
          "Sid": "VisualEditor0",
          "Effect": "Allow",
          "Action": [
              "ec2:DescribeAddresses",
              "ec2:AssociateAddress",
              "ec2:AllocateAddress"
          ],
          "Resource": "*"
      }
  ]
}
```

IAM Policy ကို ဒီအတိုင်း create လုပ်ပြီး IAM Role နဲ့ attach လုပ်ပေးရပါမယ်။ ဒါဆိုရင် IAM Role ရပါပြီ။ ရလာတဲ့ IAM Role ကို Web/App EC2 Instance မှာ Attach လုပ်ပြီး permission အလုပ်လုပ်လား မလုပ်လားစမ်းကြည့်ဖို့လိုပါတယ်။ အဲသည်တော့ EC2 Instance ထဲကို SSH Login ဝင်ပြီး **AWS CLI** နဲ့စမ်းကြည့်ပါမယ်။

```
root@ip-10-0-4-69:~# aws configure
AWS Access Key ID [None]: <- enter
AWS Secret Access Key [None]: <- enter
Default region name [ap-southeast-1]: < your region >
Default output format [json]: json
```

အရင်ဆုံး အနေနဲ့ aws cli သုံးတဲ့အခါမှာ Region မရွေးထားမိရင် error တက်နိုင်တာမို့ AWS CLI configuration လေးကို အပေါ်က Command နဲ့ Configure လုပ်ရပါတယ်။

အဲ့ဒါတွေပြီးရင်တော့ ဒီအောက်က Command လေးကို ထည့်ရိုက်ကြည့်ပါမယ်။ သူကဘာလုပ်ပေးလဲဆိုတော့ ကိုယ့်ရဲ့ EC2 Instace ID ကို AWS Metadata ကနေယူပြီးတော့ EIP ကို associate လုပ်ပေးမှာပါ။

```
INSTANCE_ID=`/usr/bin/curl -s http://169.254.169.254/latest/meta-data/instance-id`
aws ec2 associate-address --instance-id $INSTANCE_ID --allocation-id
<your_eip's_allocation_ID> --allow-reassociation
```

စမ်းလို့အဆင်ပြေတယ်ဆိုရင်တော့ ကျွန်တော်တို့က ဒီ Command လေးကို EC2 Instance စစ Boot တက်တက်ချင်းမှာ Run တဲ့ **UserData** or cloud-init မှာ သွား ထည့်ထားပေးလိုက်မယ်ဆိုရင် EC2 Auto Scaling က Instance အသစ်တက်လာတိုင်းမှာ EIP ကို Auto ချိတ်ပေးသွားမှာဖြစ်ပါတယ်။

အချုပ်အနေနဲ့ပြောရရင်တော့ ဒီ setup လေးက ဈေးသက်သက်သာသာနဲ့ ကိုယ်ရဲ့ Game Server လေးအမြဲတမ်း Up and Running ဖြစ်စေမယ့် scenario လေးတစ်ခုပဲဖြစ်ပါတယ်။ Single point of failure လည်းရှိပါသေးတယ်။ ဒါတော့ထည့်မရေးတော့ပါဘူး :D Spot Instance ကဈေးသက်သာတယ်ဆိုပေမယ့် Instance Type ရွေးချယ်တဲ့နေရာမှာလည်း နည်းနည်းစကားပြောပါသေးတယ်။ Anyway anyhow , EC2 Spot Instance ကို Auto EIP assign လုပ်ပြီး Auto Scaling Group လေးနဲ့ တွဲသုံးတဲ့ scenario လေးကတော့ ဒီလောက်ပါပဲ။ နားမလည်လိုက်တာဖြစ်စေ ထပ်ဆောင်းသိချင်လို့ဖြစ်စေ ကျွန်တော်တို့ **AWS User Group Myanmar Facebook Group** မှာ ဝင်ရောက်ဆွေးနွေးလို့ ရပါတယ်လို့ ပြောကြားရင်း ...

ဆက်လက်ကြိုးစားပါဦးမည်

ကျေးဇူးတင်လျက် AWS User Group Myanmar

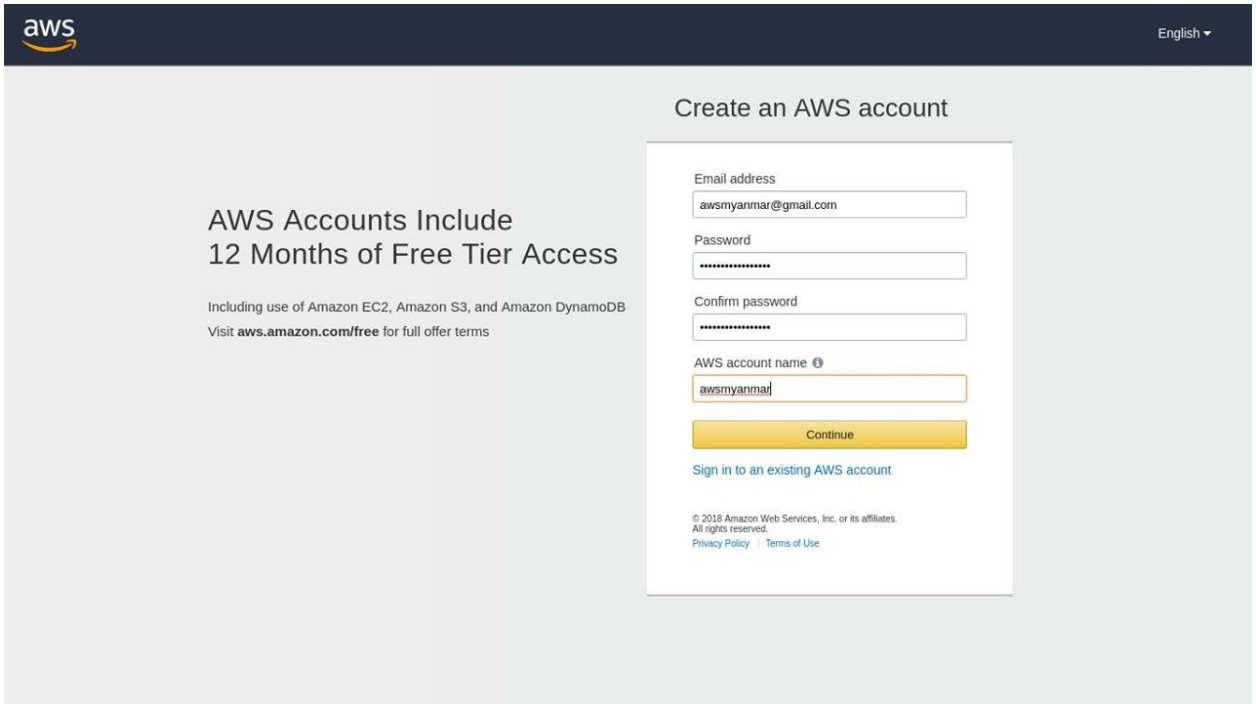
Gain free, hands-on experience with the AWS platform, products, and services

ဒီတစ်ခေါက်မှာတော့ Amazon Web Services ရဲ့ Free Tier အကောင့်ကိုဘယ်လိုဖွင့်ရမလဲ၊ Free Tier မှာဘယ်လိုမျိုး Services, Platforms, Products တွေအသုံးချလို့ရလဲ၊ အချိန်အတိုင်းအတာဘယ်လောက်ထိသုံးနိုင်မှာလဲ၊ Free Tier အကောင့်ကို အသုံးပြုပြီး ဘယ်လိုအကျိုးရှိရှိအသုံးချမယ်ဆိုတဲ့ အချက်လေးတွေကို Step-by-step အဆင့်လေးတွေကိုပြောပြပေးသွားမှာဖြစ်ပါတယ်။ လက်ရှိမှာတော့ Cloud Service Provider တွေတော်တော်များများက User တွေအနေနဲ့ အစမ်းအသုံးပြုနိုင်အောင် Credit/Debit Card ရှိရုံနဲ့ သက်ဆိုင်ရာ Cloud Service Provider တွေဆီမှာ Free Tier, $100/200 Credit ဒေါ်လာဘယ်လောက်ဖိုးသုံးစွဲခွင့်ဘာညာ အစရှိသဖြင့်ပေးထားကြပါတယ်။ ကိုယ်ပိုင် အကောင့်ရှိရုံနဲ့

အတိုင်းအတာတစ်ခုထိ Cloud Service တွေရဲ့ Hands-on lab အသုံးပြုခွင့်ရမှာဆိုတော့ Learning လုပ်ရတာပိုမိုလွယ်ကူသွားလိမ့်မယ်လို့လည်းဆိုနိုင်ပါတယ်။ ဒီတော့ စလိုက်ကြရအောင်ဗျာ၊ အရင်ဆုံးအနေနဲ့ AWS Free Tier Account ဖွင့်ဖို့ရာ Visa/Master ကဒ်လိုပါတယ်။ Visa/Master တစ်ခုခုရှိပြီးသားသူတွေအနေနဲ့ကတော့ ကဒ်ထဲမှာငွေလက်ကျန်အနည်းငယ်ရှိမယ်ဆိုရင်တော့ အဆင်ပြေပါပြီ။ Visa/Master မရှိသေးတဲ့သူတွေအတွက်ကတော့ မိမိကြိုက်နှစ်သက်ရာနီးစပ်ရာ ဘဏ်မှာ Visa/Master-Credit/Debit ဖြစ်ဖြစ်ကြိုက်နှစ်သက်ရာကဒ်တစ်ခုခုကို အရင်ဦးစွာသွားလုပ်လိုက်ပါ။ ကဒ်လုပ်ခ မြန်မာငွေ တစ်သောင်းဝန်းကျင်နဲ့ စစချင်းမှာ ကဒ်ထဲကိုတော့ USD $50 သွင်းရပါတယ်။ ဘဏ်အပေါ်မူတည်ပြီး Deposit လုပ်ရတာရှိသလို မလုပ်ရတာလည်းရှိပါတယ်။ Card လုပ်တာက ဒီနေ့လုပ်ရင် ဒီနေ့ရပါတယ် (ကြေညာမဟုတ်ပါ :D) မနက်ပိုင်းလောက်ကဒ်လုပ်ရင် ညနေပိုင်းလောက်ဆိုရင် ကဒ်ကိုအသုံးပြုလို့ရပါပြီ။ Free Tier Account ဖွင့်ဖို့လည်း Ready ဖြစ်ပြီဆိုရင် **GET AWS FREE TIRE** ကိုဝင်လိုက်ပါ။

Step-1 Create an AWS Account

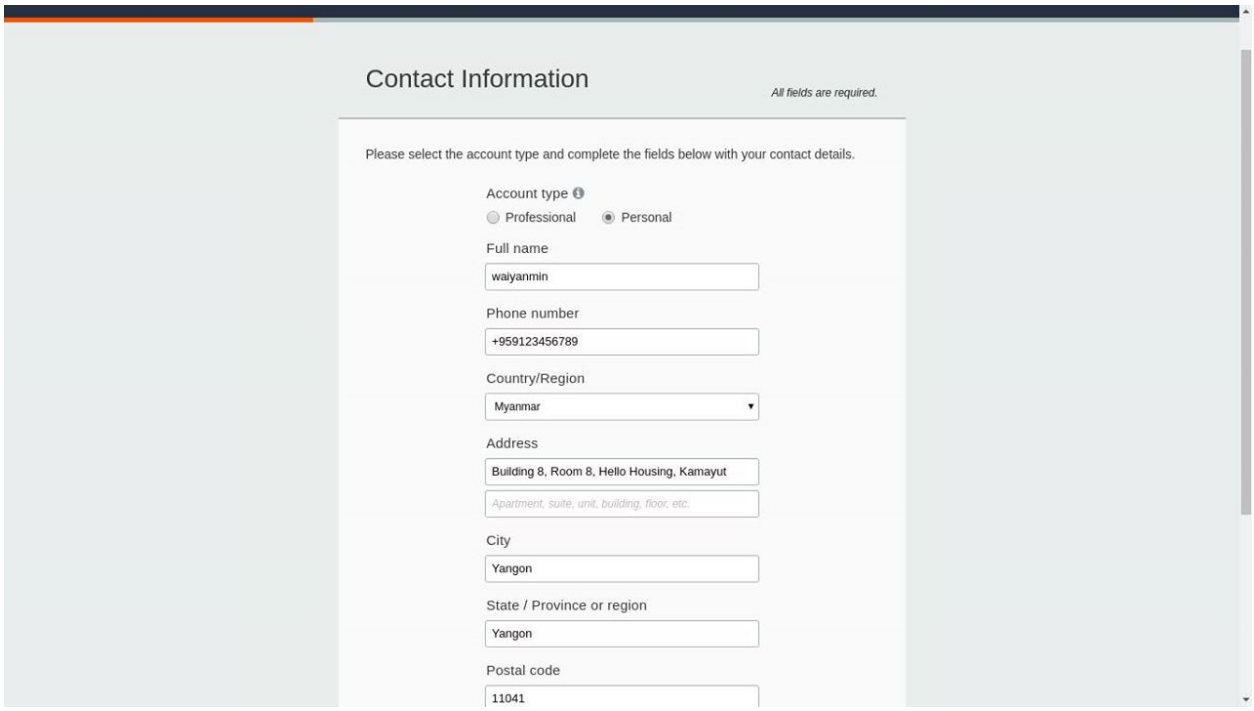
Email Address, Username, Password ဘာညာထည့်ပြီး Continue ကိုနှိပ်ပါ။



Step-2 Contact Information

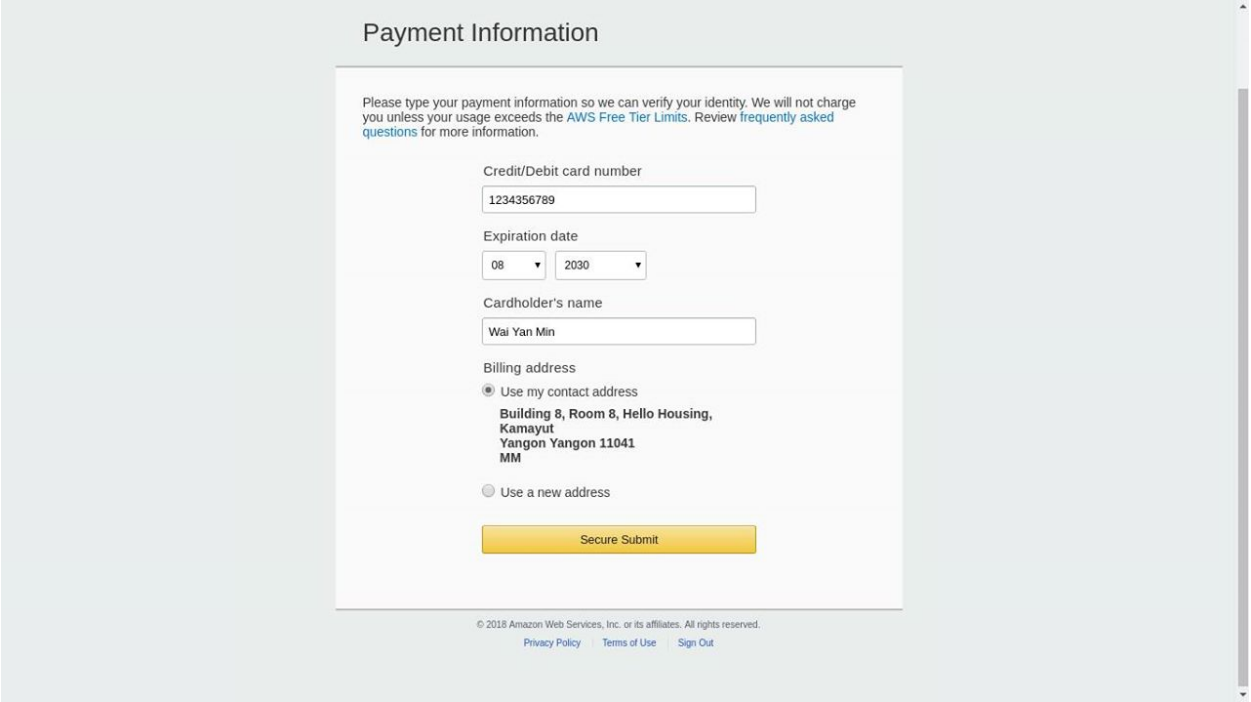
ဒုတိယအဆင့်နေနဲ့ ကိုယ့်ရဲ့ Contact Information ကိုထည့်ပေးရပါမယ်။ ဒီနေရာမှာတစ်ချက်ရှင်းပြပါရစေ။ Contact Information ဖြည့်ခိုင်းတဲ့နေရာ အရင်ဆုံး *Account Type* ကိုရွေးပေးရပါမယ်။ Professional & Personal ဆိုပြီး ၂ ခုရှိပါတယ်။ Professional Account Type

ကတော့ မိမိအနေနဲ့ AWS ရဲ့ Service တွေကို Individual Company အနေနဲ့ အသုံးပြုမယ်ဆိုရင်တော့ Professional Account Type ကိုရွေးချယ်သင့်ပါတယ်။ ဒီလိုမှမဟုတ်ဘူး SystemAdmins/Developers/EndUsers တွေက မိမိအနေနဲ့ တစ်ကိုယ်ရည်တစ်ကာယသော်လည်းကောင်း၊ မိမိ Project အတွက်သော်လည်းကောင်း အသုံးပြုမယ်ဆိုရင်တော့ Personal Account Type ကိုရွေးချယ်သင့်ပါတယ်။ ဒီ Account Type ၂ ခုက Functionality အရ ထွေထွေထူးထူးကွာခြားချက်ရယ်လို့မရှိပါဘူး။ Professional Account Type ကိုရွေးမယ်ဆိုရင်တော့ ကိုယ်ရဲ့ Company Name ထည့်ခိုင်းပါမယ်။ Personal Account Type ရွေးချယ်မယ်ဆိုရင်တော့ Company Name ထည့်စရာမလိုပါဘူး။ Contact Information ကိုပြည့်စုံအောင်ဖြည့်ပြီးပြီဆိုရင်တော့ Customer Agreement ကို Mark လုပ်ပြီး နောက်တစ်ဆင့်ကို ဆက်လက်လုပ်ဆောင်ပါမယ်။



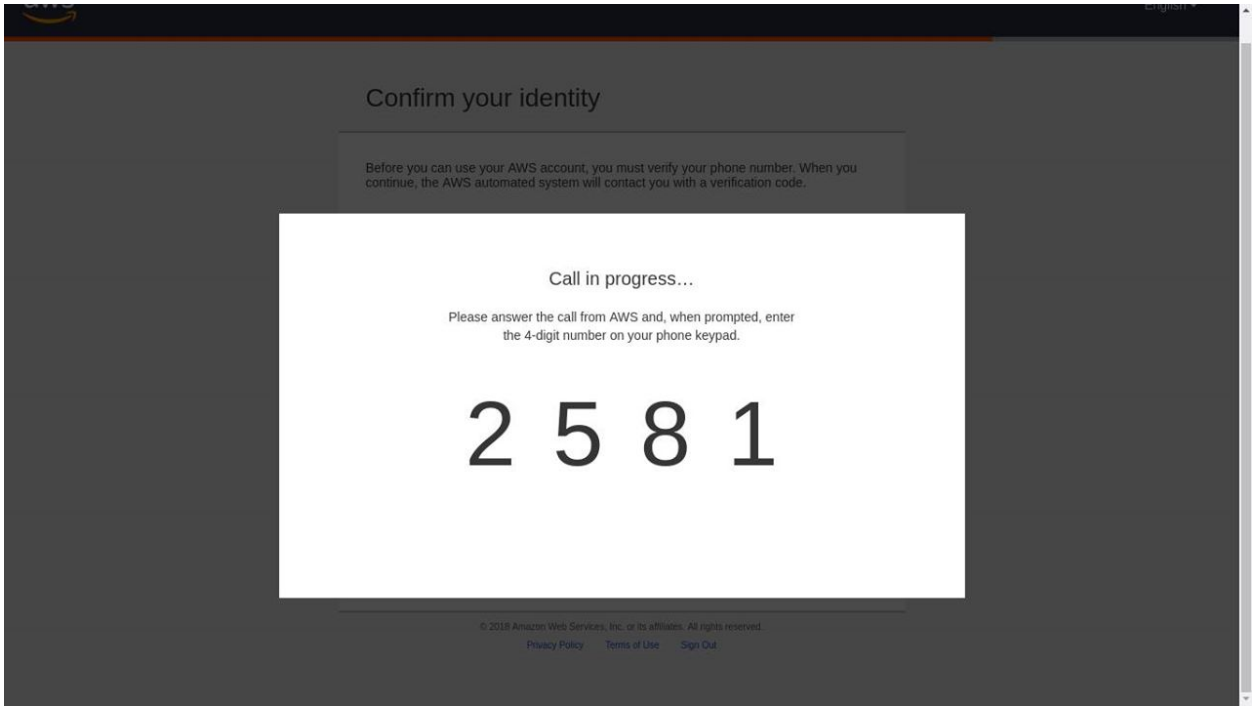
Step-3 Payment Information

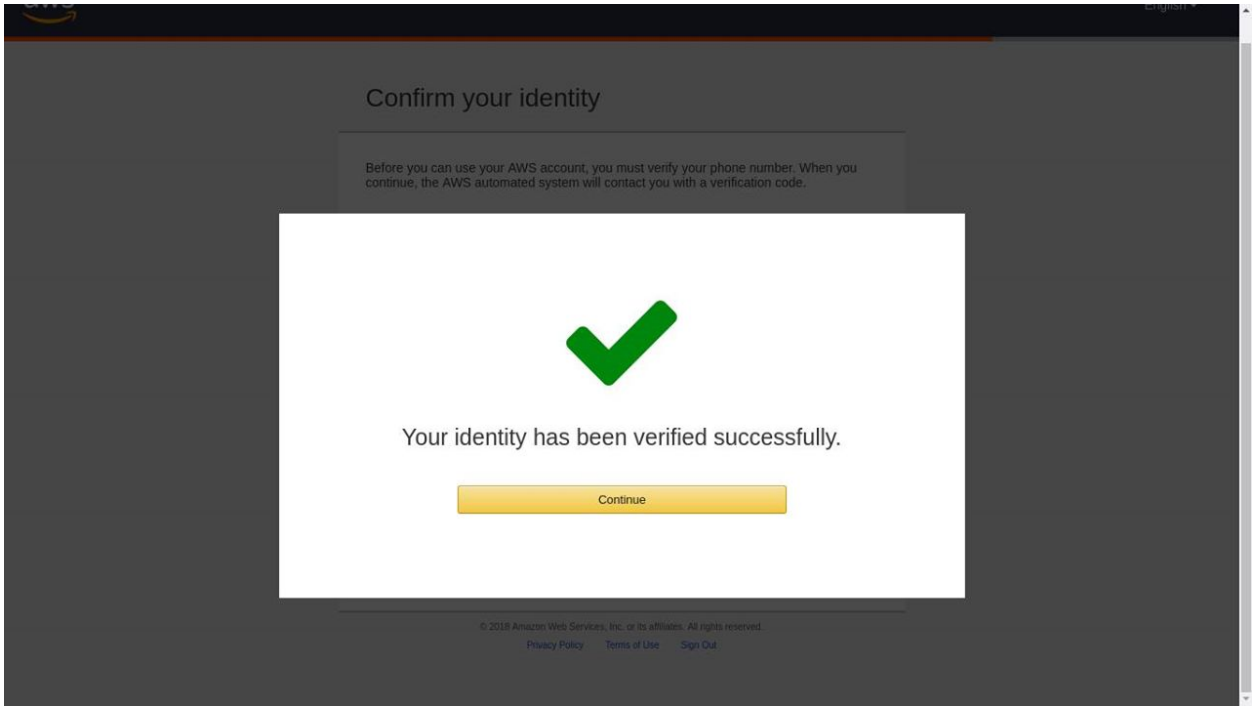
တတိယအဆင့် ကတော့ AWS အနေနဲ့ Free Tier Usage အတွက် မိမိဆီက Payment Information ကိုတောင်းပါလိမ့်မယ်။ ကျွန်တော်တို့ စစချင်းမှာပြောခဲ့သလိုပဲ Credit/Debit card number တစ်ခုခုနဲ့ Link လုပ်ပေးရပါမယ်။ ကဒ်မှာပါတဲ့ Information လေးတွေကို သေချာဂရုတစိုက်ဖြည့်သွင်းပါ။ နောက်ပြီးရင်တော့ Billing Address မှန်၊ မမှန်ပြန်စစ်ပြီး Secure Summit ကိုနှိပ်လိုက်ရင်အဆင်ပြေပါပြီ။



Step-4 Confirm you Identity

ဒီအဆင့်မှာတော့ မိမိ Account Confirmation အတွက် Active (လက်ရှိအသုံးပြုနေတဲ့) ဖုန်းနံပါတ်တစ်လုံးကိုထည့်ပေးရပါမယ်။ AWS အနေနဲ့ Direct Phone Call မှတစ်ဆင့် 4-Digit Code တစ်ခု ကိုပြောပြီး Account ဖွင့်သူအနေနဲ့ Direct Phone Call မှတစ်ဆင့်ပြောသော 4-Digit Code ကိုပြန်လည်ထည့်သွင်းပေးရမှာဖြစ်ပါတယ်။ 4-Digit Code ကိုမှန်ကန်အောင်ဖြည့်ပြီးပြီဆိုရင်တော့ Your account has been verified successfully ဆိုပြီးပေါ်မှာလာမှာ။ ထို့နောက် Continue ကို နှိပ်ကာ **Final Step** ကိုရောက်ပါမယ်။

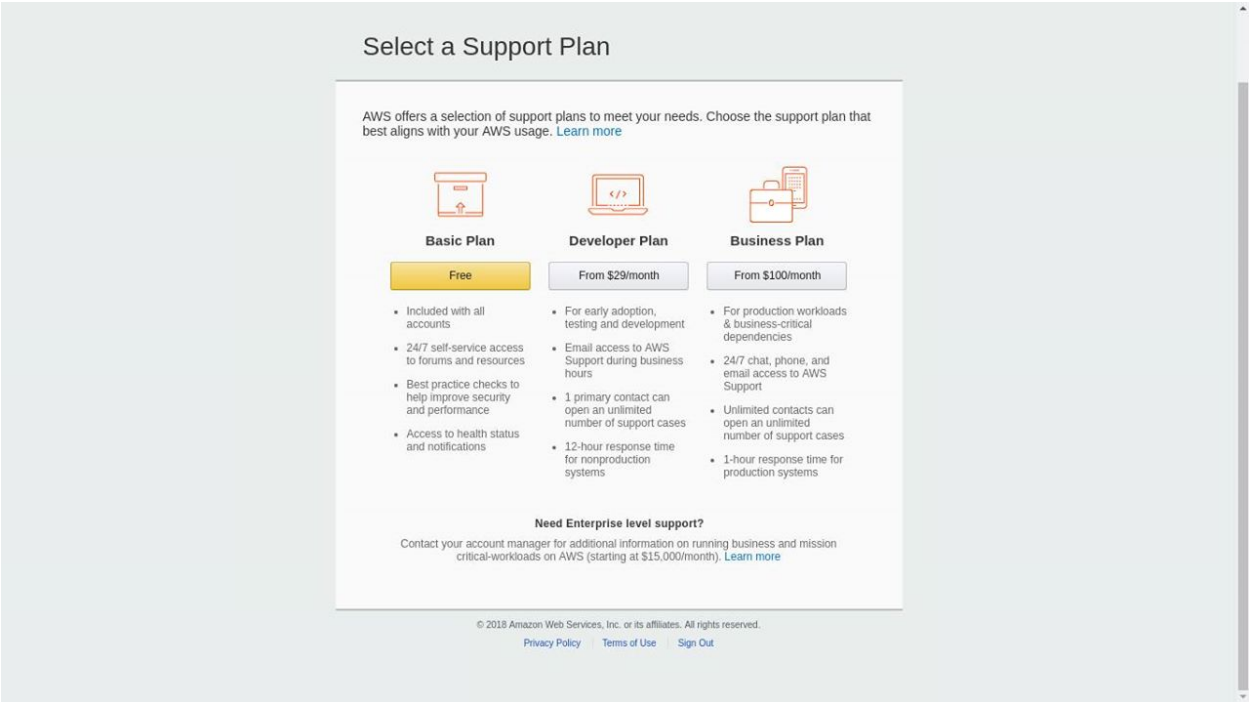




Step-5 Final Step

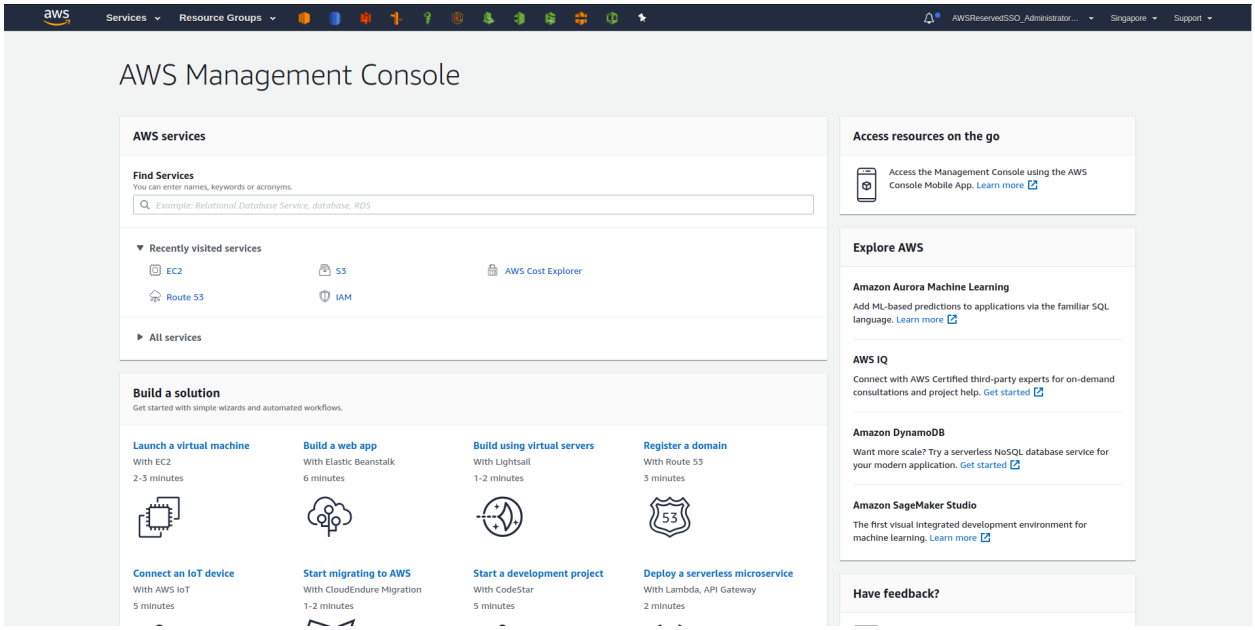
Final Step မှာတော့ ကျွန်တော်တို့အနေနဲ့ မိမိတို့အကောင့်ကိုဘယ်လို Support Plan တွေနဲ့သုံးမလဲဆိုတာကိုရွေးလို့ရပါမယ်။ အခု အကောင့်က Company အတွက်သုံးတာမျိုးမဟုတ်ဘူး

AWS ရဲ့ Support တွေဘာမှမလိုဘူးဆိုရင်တော့ Basic Plan လောက်နဲ့တင်အဆင်ပြေပါတယ်။ ဒီလိုမှမဟုတ်ဘူး ကိုယ့်အနေနဲ့ တစ်ခုခုဖြစ်ရင် AWS ရဲ့ Support တွေကိုမြန်မြန်ဆန်ဆန်လိုအပ်တယ်ဆိုရင်တော့ Developer Plan နဲ့ Business Plan ကိုရွေးချယ်လို့ရပါတယ်။ AWS ရဲ့ Support Plan ကွာခြားချက်တွေကတော့ အောက်ကပုံမှာကြည့်နိုင်ပါတယ်။



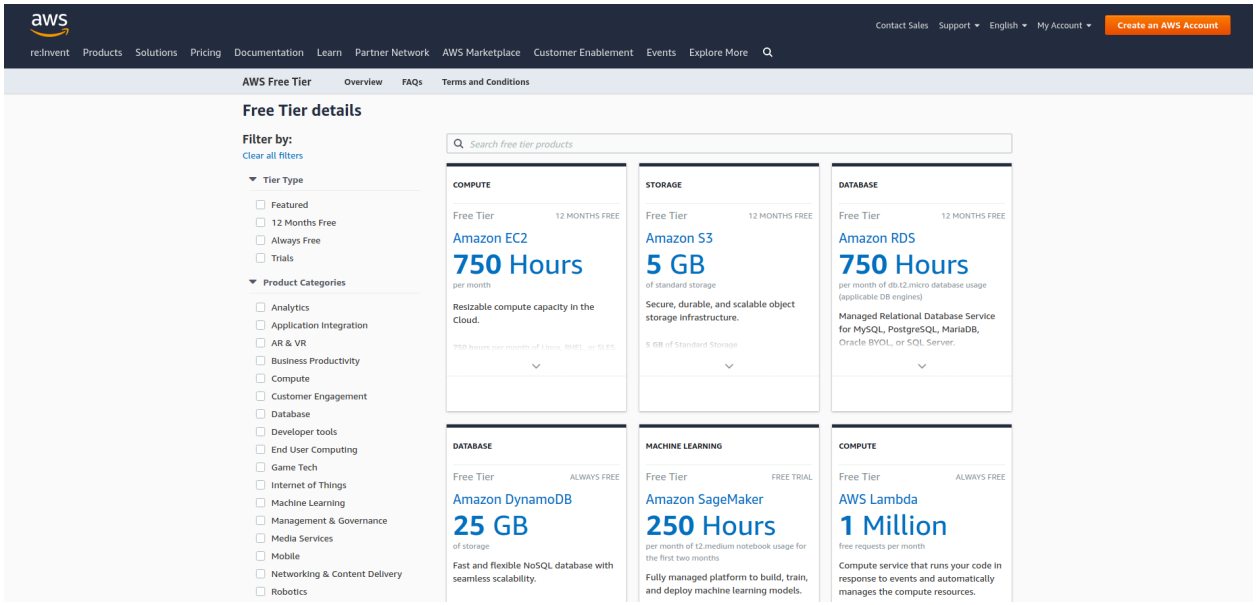
Support Plan

Support Plan ကိုရွေးချယ်ပြီးရင်တော့ ~ Tada ~ AWS Web Management Console ကနေ မိမိကြိုက်နှစ်သက်ရာ Services တွေကို စတင်စမ်းသပ် အသုံးပြုနိုင်ပါပြီခင်ဗျာ။



AWS Management Console

မှတ်ချက်အနေနဲ့ကတော့ AWS က Free Tier ကို အကောင့်စဖွင့်တဲ့နေ့မှစပြီး ၁၂ လ (၁နှစ်) တိတိ Limited Resource တွေကိုအသုံးပြုခွင့်ရပါမယ်။ Free Tier မှာ ဘယ်လောက်အတိုင်းအတာထိအသုံးပြုလို့ရမယ်ဆိုတာကို ဒီလင့်ခ် မှာကြည့်နိုင်ပါတယ်။ Limited ဆိုပေမယ့် အတော်လေးကိုအဆင်ပြေပါတယ်။ ဥပမာဆိုရင် ကျွန်တော်တို့ အသုံးများကြတဲ့ Elastic Compute Cloud (EC2) Instance တစ်လုံးကို တစ်လမှာ နာရီ ၇၅၀ သုံးဆွဲခွင့်ပြုထားပါတယ်။ နာရီ ၇၅၀ ဆိုတဲ့ အချိန်က ရက်ပေါင်း ၃၁ ရက် ရှိမဲ့လ တွေမှာဆိုရင်တောင်မှ ကောင်းကောင်းကြီး Run ဖို့လုံလောက်တဲ့ အချိန်ပမာဏဖြစ်ပါတယ်။ အကောင့်ဖွင့်ထားပြီး ၁၂ ကျော်တဲ့ အကောင့်တွေကတော့ Free Tier အသုံးခွင့်ရှိတော့မှာမဟုတ်ပါဘူး။



Free Tier Limit

ဟုတ်ကဲ့။ အခုလောက်ဆိုရင်တော့ AWS Free Tier Account ကို
အလွယ်တကူဖွင့်လို့ရမယ်လို့ယုံကြည်ပါတယ် ခင်ဗျာ။ နောက်လာမည့် အပတ်မှာတော့ Things to do after registering AWS free tier account ဆိုတဲ့ ဆောင်ရန်ရှောင်ရန်လေးများအကြောင်းကို ဆက်လက်ရေးသွားဦးမှာမို့ အားပေးကြပါဦးခင်ဗျာ။ ဆက်လက်ကြိုးစားပါဦးမည်

Using AWS Systems Manager to login EC2 instance without SSH

AWS Ec2 instance တစ်လုံး စဆောက်လိုက်ပီးတာနဲ့.အဓိကပထမဆုံးလုပ်တာက ဆောက်ထားတဲ့ instance ထဲLogin ဝင်ပီး Manage လုပ်တာပါပဲ။ ဆိုတော့ Login ဝင်မယ်ဆို အဓိက Windows ဆို RDP Protocol , Linux ဆို SSH နဲ့ဝင်ကြရပါတယ်။ခုtutorialလေးက Linux instance ကို လုပ်ပြမှာဆိုတော့ SSH ပေါ့။

AWS Ec2 Linux instance တစ်လုံးကို စတင်ဝင်ရောက်ဖို.ဆိုရင် 1. Pem Key (Private Key) တစ်ခု 2. AWS Security Group ကနေ SSH Port 22 ကို Allow လုပ်ပေးဖို.လိုအပ်ပါတယ်။

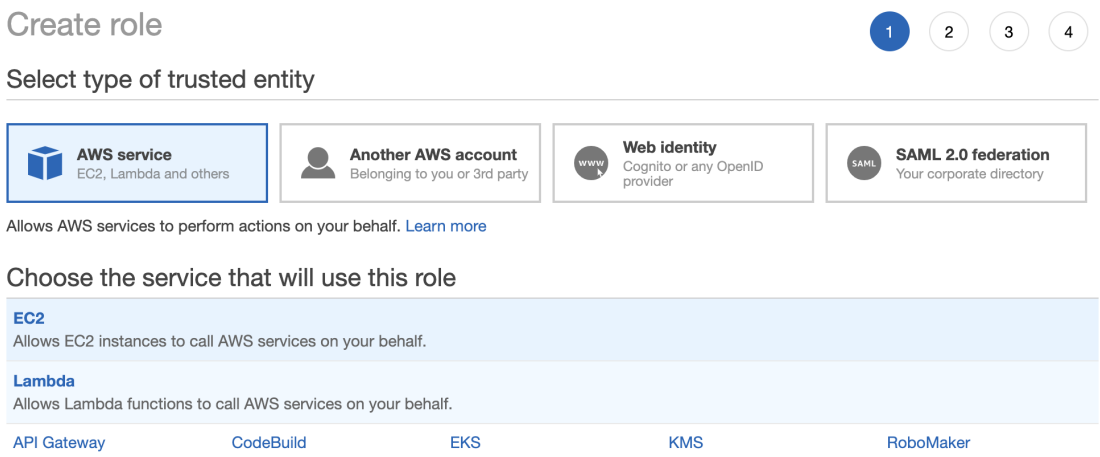
ခုလုပ်ပြမှာလေးက AWS Ec2 instance တစ်လုံးကို Login ဝင်တဲ့အခါ SSH မလိုပဲ AWS ရဲ့ Service တစ်ခုဖြစ်တဲ့ Systems Manager, Session Manager ကနေ Login ဝင်တဲ့ နည်းလမ်းလေးဖြစ်ပါတယ်။SSH မသုံးပဲ ဘာလို. AWS Systems Manager, Session Manager ကိုသုံးလဲ မေးစရာရှိလာပါတယ် ဘာလို့.သုံးလဲ ဘာကြောင့်လဲဆိုရင် အဓိက အားဖြင့် ကျနော်အမြင်အရကော Blogs တွေမှာပြောထားတာက -

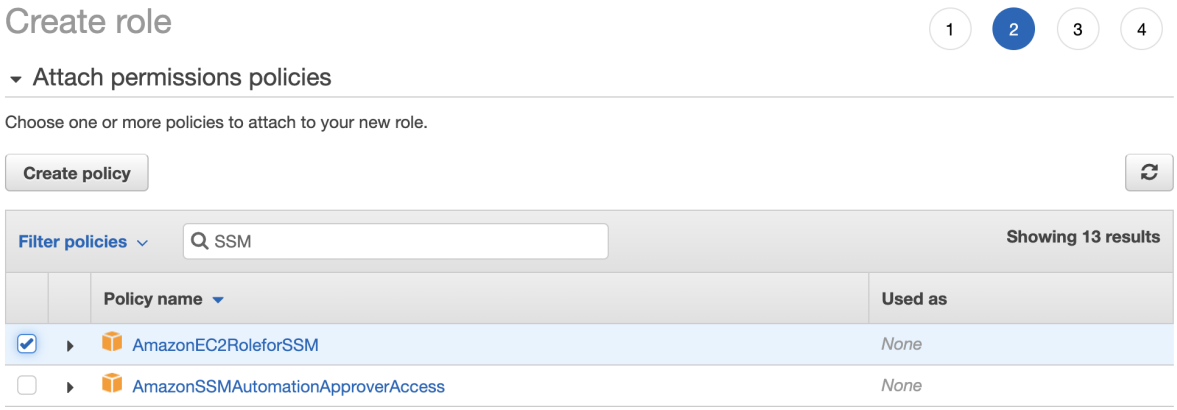
- SSH bastion Host မလိုတော့ဘူး။ (ဈေးသက်သာတာပေါ့နော်)
- SSH Key မလိုတော့ဘူး။
- No need to allow SSH in AWS Security Group (AWS Security Group မှာ SSH Allow ဖို.မလိုတော့ဘူး)(အဲ့တာလေးတော့ကြိုက်တယ်)
- နောက်ဆုံးတစ်ခုကတော့ Logs တွေကို AWS Cloudwatch Log groups ကနေ Default ကြည့်လို.ရတာပါပဲ။

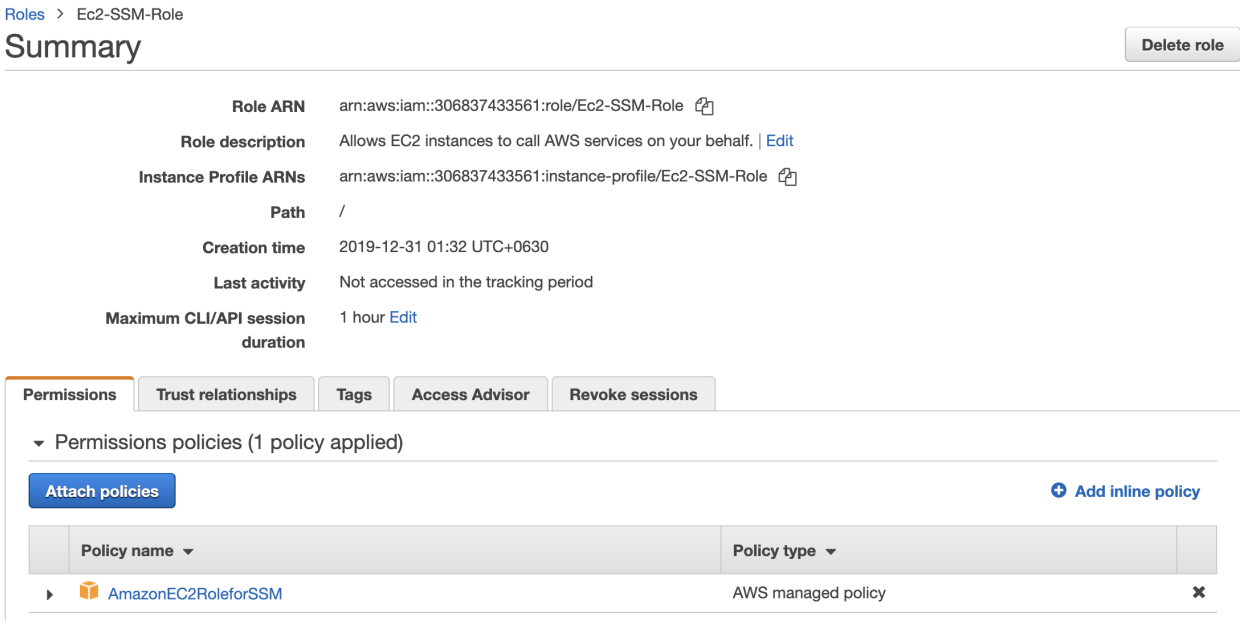
အိုကေ စကြည့်လိုက်ရအောင် ...

ပထမဆုံးဘာမှ မစခင် ကျနော်တို. Systems Manager, Session Manager တွက် IAM Ec2 Role တစ်ခုဆောက်ပေးဖို.လိုပါတယ် (Systems Manager ကနေ instance ကို access ရဖို.တွက်)

ကျနော်တို.အောက်ကပုံမှာပြထားတဲ့အတိုင်း IAM Console ကနေ Role တစ်ခု create လုပ်ပါမယ်။







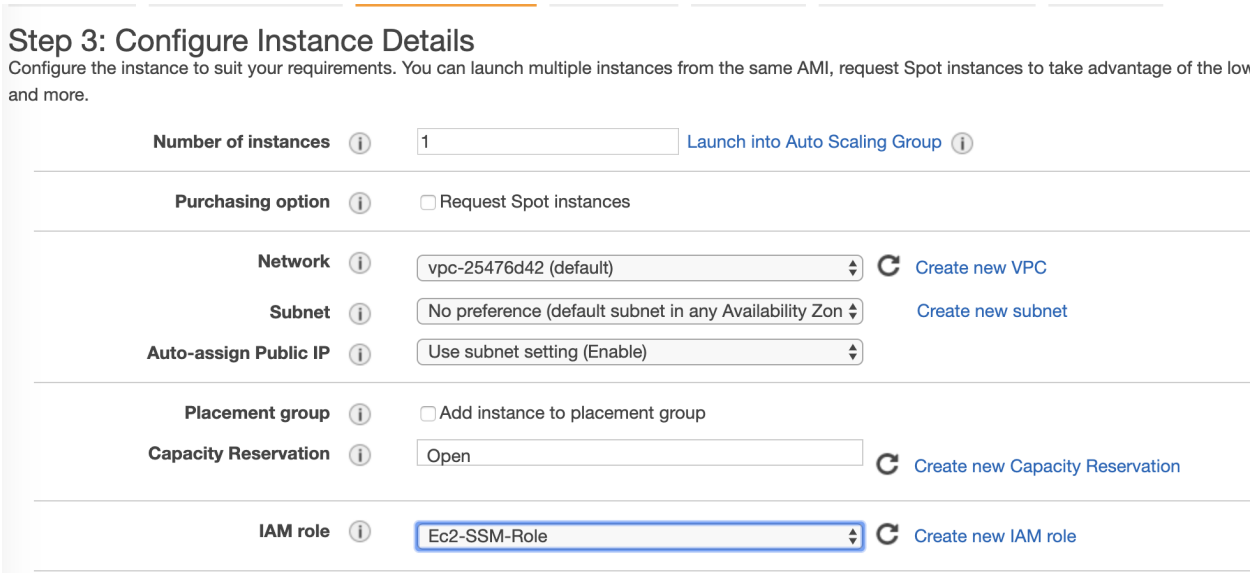
ထို.နောက် IAM Role Create လုပ်ပီးပီ ဆိုတာနဲ့ ကျနော်တို. Ec2 instance ဆောက်ကြရအောင်.မဆောက်ခင် Ec2 Linux Instance မှာ SSM Agent Install

လုပ်ထားဖို.လိုပါတယ်.Amazon Linux AMI တွေမှာ တော့ Default install လုပ်ပြီးသားပါ တခြား Linux AMI တွေမှာ တော့ သွင်းပေးရပါမယ်.အောက်မှာဖော်ပြထားတဲ့ link မှာကြည့်ရှုနိုင်ပါတယ်။

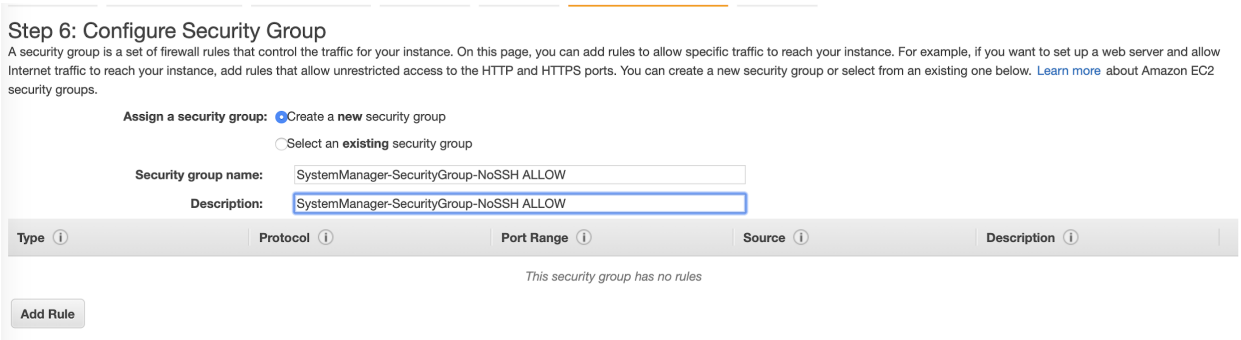
https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-install-ssm-agent.html

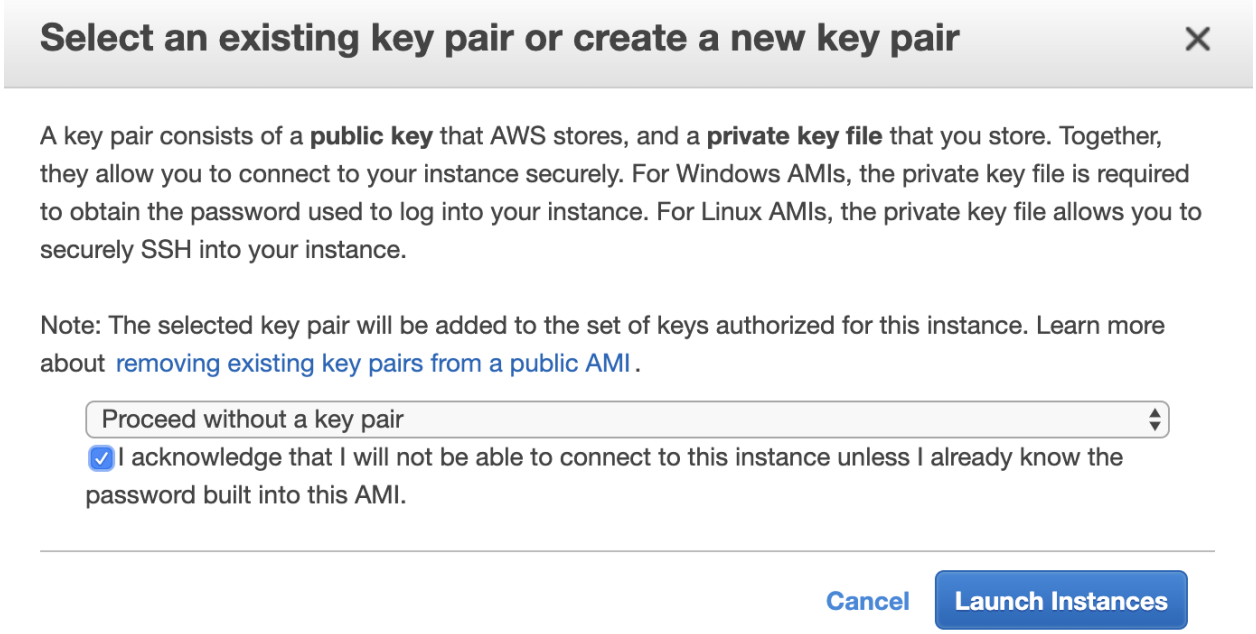
အခုကျွန်တော်က Amazon Linux AMI နဲ့ဆောက်ပြမှာဆိုတော့ ssm agent က default သွင်းပြီးသားပါလာပါလိမ့်မယ်။

ကျွန်တော်တို. IAM Role နေရာမှာခုနက create လုပ်ထားတဲ့ Role ကိုရွေးချယ်ပေးရပါမယ်။

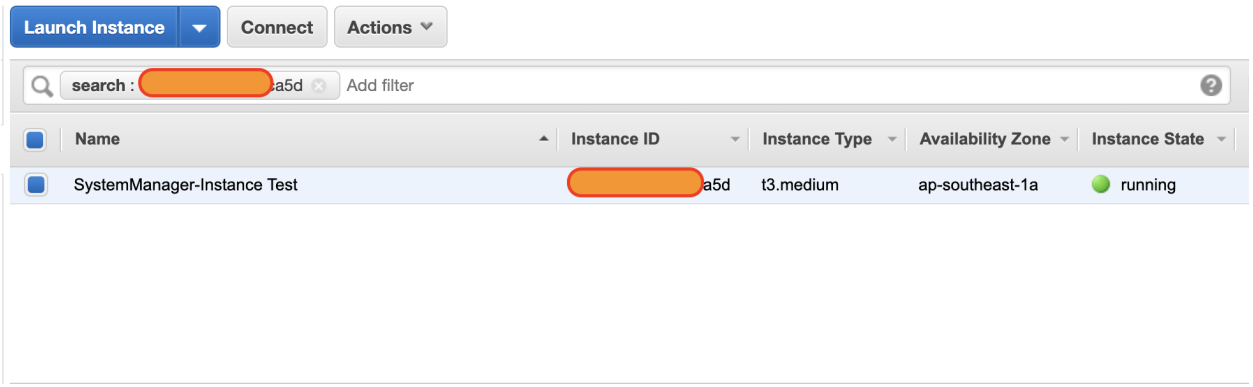


ထို.နောက်အပေါ်မှာ ပြောထားသလို Security Group မှာ ဘာကိုမှ Allow မလုပ်ထားပါဘူး..SSH Key Pair နေရာမှာလဲ ဘာ Key မှ မထည့်ထားပါဘူး။

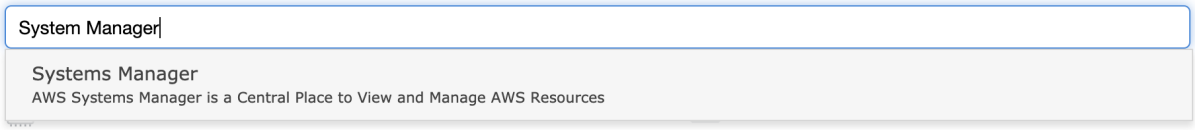




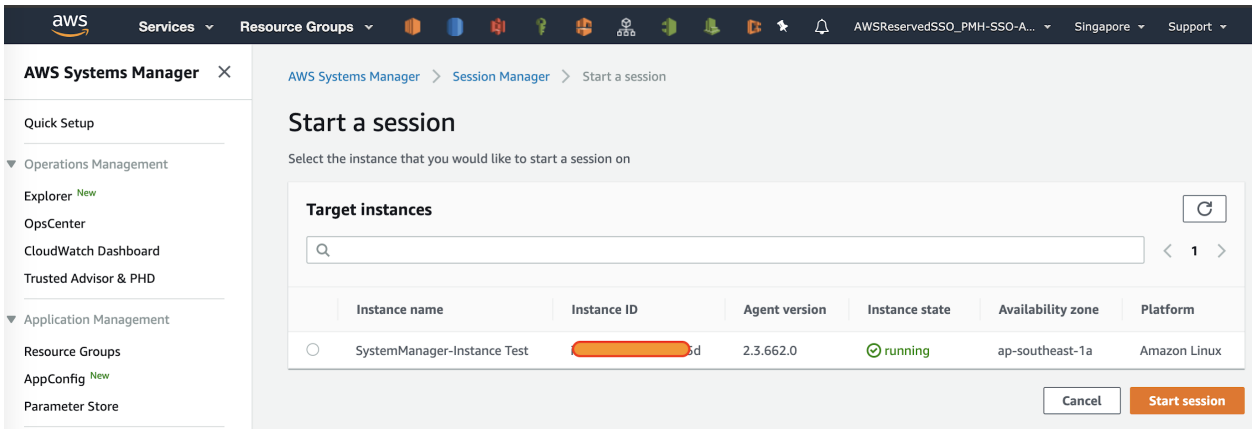
နောက်ဆုံးကျနော်တို့. Ec2 instance တစ်ခုကို အောင်မြင်စွာဆောက်လို့.ရပါပီဗျာ။



ထို.နောက် ကျွန်တော်တို့. AWS Systems Manager ကိုသွားပါမယ်။ကျနော်တို့.Ec2 instance ပေါ်နေပီလားသိရအောင် ...



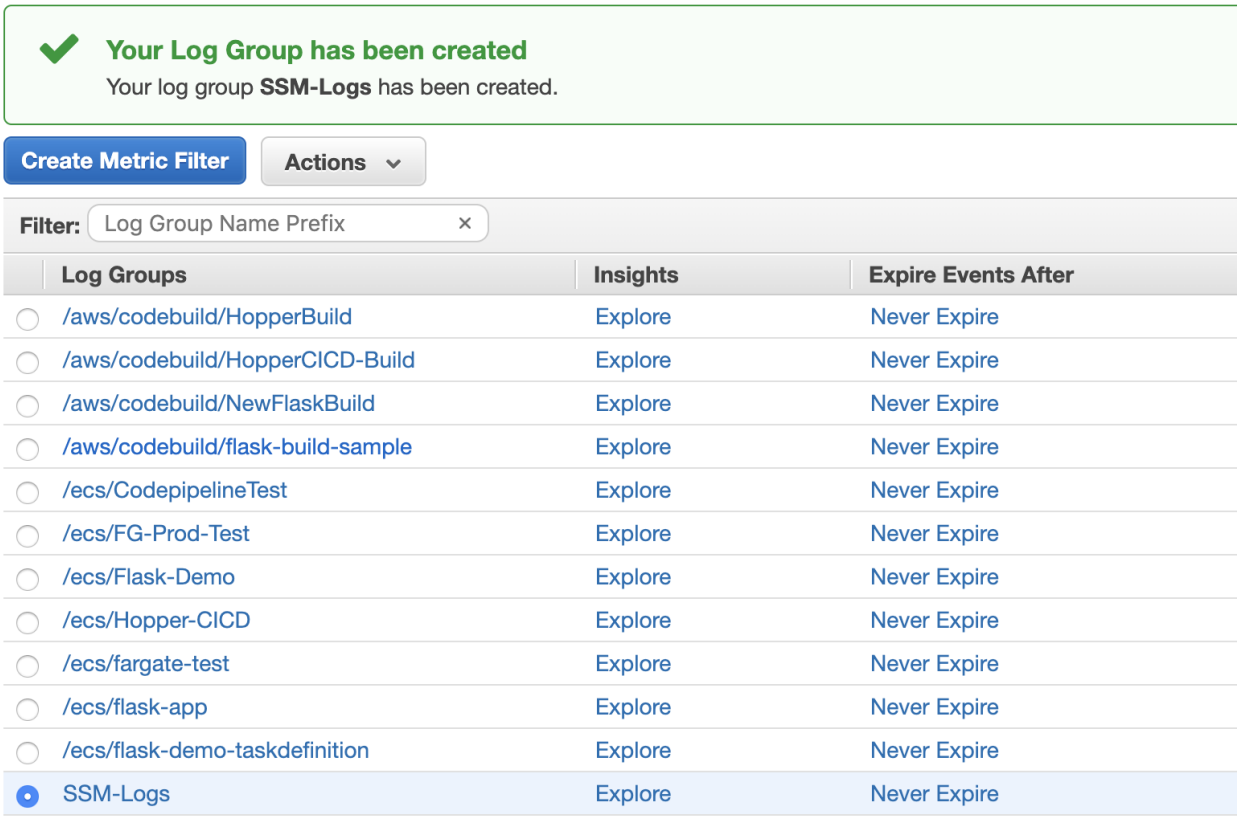
Instance ကို မြင်နေရပါပီဗျာ ...



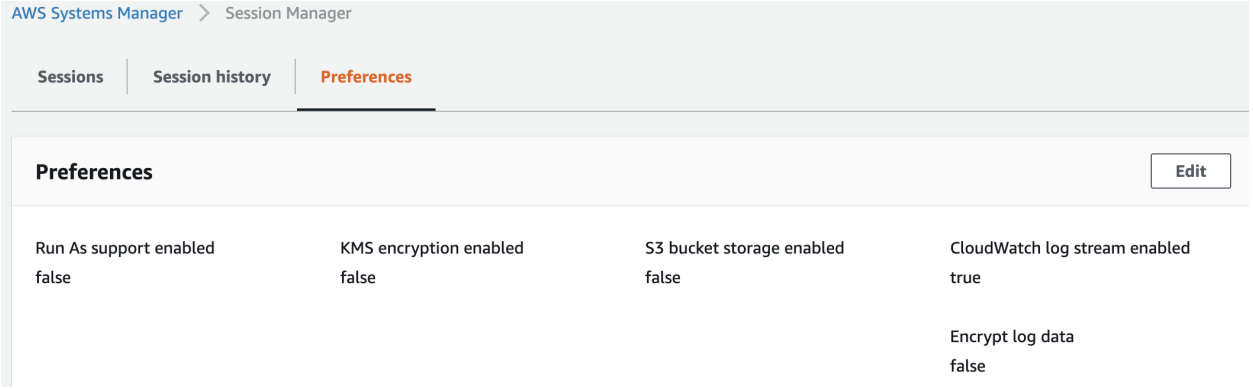
အိုကေ ကျနော်တို့. Session open ပီး instance ထဲ Login ကြည့်ကြရအောင် ...

Systems Manager ကနေဝင်လို.ရပါတယ်။ဘာ Pem Key မှမလိုသလို ဘာ Security Group မှာမှ Allow ပေးထားစရာမလိုပါဘူး။ဆိုတော့ကျွန်တော်တို.လုပ်တာ အကုန်မှန်တယ်ပေါ့.. ဒါပေမဲ့ ကျနော်တို.နောက်ဆုံးအနေနဲ့ Logs လုပ်ကြည့်ဖို.လိုပါသေးတယ် AWS CloudWatch ကနေကျွန်တော်ကတော့ ဘာတွေလုပ်သွားလည်းသိဖို.တွက် Logs ကိုလုပ်တာပါ။

Logs တွေကိုထုတ်မယ်ဆို ကျွန်တော်တို. AWS Cloudwatch ကို သွားပါမယ်။ CloudWatch Console ကိုရောက်ရင် ***log groups*** မှာ လိုချင်တဲ့ နာမည်နဲ့ log group တစ်ခုကို create လုပ်ပါမယ်။ကျွန်တော်ကတော့ ***SSM-Logs*** ဆိုပီး နာမည်နဲ့တစ်ခုဆောက်လိုက်ပါမယ်။



ထို.နောက် Systems Manager Console ကိုပြန်သွားပီး Log group ကို Setting ချပါမယ်။



Send session output to CloudWatch Logs

You can stream and encrypt log data for all sessions in your account to a CloudWatch Logs log group in your account. **Learn more**

- [x] CloudWatch logs
- [ ] Encrypt log data

CloudWatch log group

Enter the name of the log group to upload session logs to.

- (●) Choose a log group name from the list
- ( ) Enter a log group in the text box

CloudWatch log groups

SSM-

1 **2**

	Log groups	Encryption	Expire events after	Metric filters	Stored bytes	Creation time
( )	/ecs/flask-demo-taskdefinition	Not encrypted	Never expire	0	6160	Fri, 22 Nov 2019 04:33:34 GMT
(●)	SSM-Logs	Not encrypted	Never expire	0	0	Mon, 30 Dec 2019 19:51:32 GMT

အထက်ပါပုံအတိုင်း setting ချပီးရင် Save မှတ်ပါမယ်။နောက်ဆုံးအနေနဲ့ Session Login ပြန်ဝင်ပီး Linux မှာ Yum Update နဲ့ တခြား action လုပ်ကြည့်ကြရအောင်..

ပထမဆုံးအနေနဲ့ ကျွန်တော်တို. Yum Update နဲ. Stress Application လေးကိုသွင့်ကြည့်ရအောင်

```
otal download size: 49 M
ownloading packages:
elta RPMs disabled because /usr/bin/applydeltarpm not installed.
1/17): ca-certificates-2018.2.22-70.0.amzn2.0.1.noarch.rpm                                  | 392 kB
2/17): file-5.11-35.amzn2.0.2.x86_64.rpm                                                    |  57 kB
3/17): file-libs-5.11-35.amzn2.0.2.x86_64.rpm                                               | 339 kB
4/17): amazon-ssm-agent-2.3.714.0-1.amzn2.x86_64.rpm                                        |  15 MB
5/17): krb5-libs-1.15.1-37.amzn2.2.1.x86_64.rpm                                             | 759 kB
6/17): libidn2-2.3.0-1.amzn2.x86_64.rpm                                                     | 140 kB
7/17): python-2.7.16-4.amzn2.x86_64.rpm                                                     |  92 kB
8/17): python-devel-2.7.16-4.amzn2.x86_64.rpm                                               | 403 kB
9/17): kernel-4.14.154-128.181.amzn2.x86_64.rpm                                             |  20 MB
10/17): python-libs-2.7.16-4.amzn2.x86_64.rpm                                               | 7.4 MB
11/17): python2-rpm-4.11.3-40.amzn2.0.3.x86_64.rpm                                          |  84 kB
12/17): rpm-4.11.3-40.amzn2.0.3.x86_64.rpm                                                  | 1.2 MB
13/17): rpm-build-libs-4.11.3-40.amzn2.0.3.x86_64.rpm                                       | 106 kB
14/17): rpm-libs-4.11.3-40.amzn2.0.3.x86_64.rpm                                             | 275 kB
15/17): rsyslog-8.24.0-41.amzn2.2.1.x86_64.rpm                                              | 613 kB
16/17): yum-3.4.3-158.amzn2.0.3.noarch.rpm                                                  | 1.2 MB
17/17): rpm-plugin-systemd-inhibit-4.11.3-40.amzn2.0.3.x86_64.rpm                           |  47 kB
--------------------------------------------------------------------------------------------------------------
otal                                                                                 90 MB/s |  49 MB
unning transaction check
unning transaction test
ransaction test succeeded
unning transaction
 Updating   : rpm-4.11.3-40.amzn2.0.3.x86_64
 Updating   : rpm-libs-4.11.3-40.amzn2.0.3.x86_64
 Updating   : python-libs-2.7.16-4.amzn2.x86_64
 Updating   : python-2.7.16-4.amzn2.x86_64
 Updating   : file-libs-5.11-35.amzn2.0.2.x86_64
 Updating   : rpm-build-libs-4.11.3-40.amzn2.0.3.x86_64
 Installing : python2-rpm-4.11.3-40.amzn2.0.3.x86_64
 Updating   : yum-3.4.3-158.amzn2.0.3.noarch
```

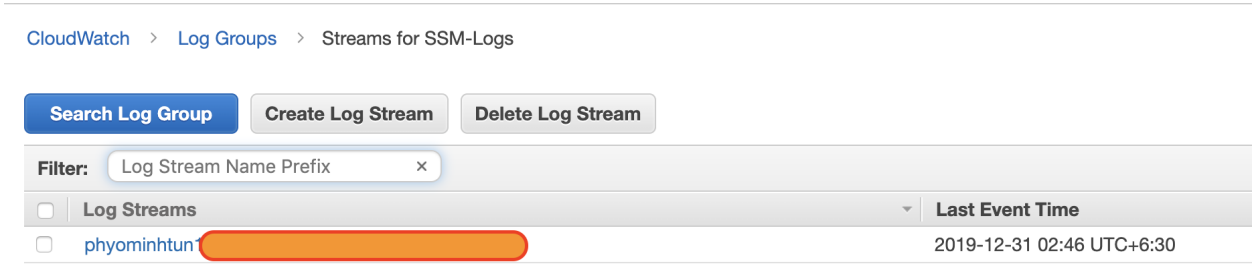
```
[root@ip-1[redacted]2 ~]# yum install stress
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
188 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package stress.x86_64 0:1.0.4-16.el7 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================
 Package                                Arch                                Version
================================================================================
Installing:
 stress                                 x86_64                              1.0.4-1

Transaction Summary
================================================================================
Install  1 Package

Total download size: 39 k
Installed size: 94 k
Is this ok [y/d/N]:
```

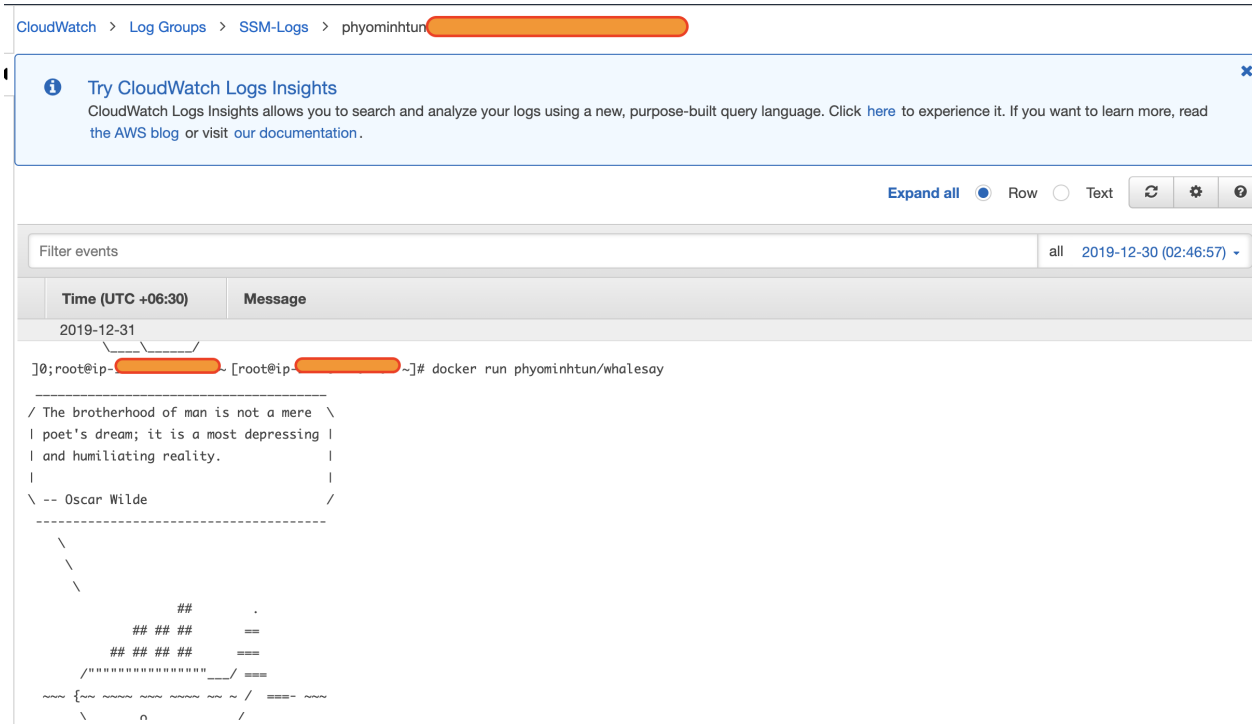
ဘာညာနည်းနည်းပါးပါ လျောက် Run ကြည့်မယ်။

```
[root@ip-1[redacted]2 ~]# docker run phyominhtun/whalesay
 _____________________________________
/ Stability itself is nothing else than a \
\ more sluggish motion.                  /
 -------------------------------------
    \
     \
      \
                    ##        .
              ## ## ##       ==
           ## ## ## ##      ===
       /""""""""""""""""___/ ===
  ~~~ {~~ ~~~~ ~~~ ~~~~ ~~ ~ /  ===- ~~~
       \______ o          __/
        \    \        __/
          \____\______/
```

ထို.နောက် AWS Cloudwatch Log မှာပေါ်ဖို.စောင့်ရပါလိမ့်မယ်။ချက်ချင်းကြီးတော့မရောက်သေးပါဘူး ခဏတော့စောင့်ပေးရပါတယ်။

ငါးမိနစ်လောက်စောင့်လိုက်ရင် log တွေ Cloudwatch log group ထဲဝင်လာပီး Session event logs များကိုကြည့်လို.ရပါပီ။





နိုဂုံးချုပ်ရမယ်ဆိုရင်တော့ဖြင့် မိမိ Ec2 Instance ကို SSH နဲ့သာမက AWS Systems Manager ရဲ့ Session Manager နဲ့လဲ access and management လုပ်နိုင်ပါတယ်၊တစ်ခုနဲ့တစ်ခုမတူတာလေးတွေ အားသာချက်အားနည်းချက်လေးတွေလိုက်ပီး ကိုယ့် usecase အပေါ်မူတည်ပြီးအသုံးပြုနိုင်ပါတယ်။ အခု tutorial လေးကိုနောက်နေ့မှာ ကျွန်တော်တို. Video Record လေးနဲ့ လုပ်နည်းလေးတင်ပေးပါဦးမယ်။

Managing private EC2 instance with Bastion Host

Bastion Hosts on AWS Cloud

ကျွန်တော်တို. AWS မှာ Infrastructure စဆောက်ရင် VPC ဆောက်ကြပီးတော့ public subnet တွေ private subnet တွေ စခွဲကြပါတယ်။ပြီးတော့ route table နဲ့ route တွေခွဲ ပြီးတော့ NAT gateway or NAT instance setup လုပ်ကြနဲ့စတင်လုပ်ကြရပါတယ်။ အဲ့မှာ ကျွန်တော်ဆို public subnet မှာ ဆို loadbalancer , NAT gateway or instance ထားပြီး တကယ် run နေတဲ့ instance တွေ DB တွေဆိုတတ်နိုင်သလောက် private subnet မှာ ထားပါတယ်။ Security အရဆိုလည်း ကောင်းတာပေါ့။

ဥပမာ Web Server တစ်လုံးကို VPC Private Subnet မှာထားပြီး ALB or Classic Load Balancer public subnet မှာထားပြီး run ဖြစ်ပါတယ်။ ဒါဆို user တွေက loadbalancer endpoint (or) AWS Route 53 DNS name နဲ့ လှမ်း Access လုပ်လို.ရပါတယ်။နောက်ပြီး public subnet မှာရှိတဲ့ instance

တွေ့ဆို SSH or RDP နဲ့လှမ်း access လုပ်ဖို့.အဆင်ပြေပေမဲ့ private subnet မှာရှိတဲ့ instance တွေ့ဆို လှမ်း access လုပ်ဖို့.နည်းနည်းအခက်အခဲ ရှိလာပါတယ်။

ဒါဆို ဘယ်လိုအခြေနေမှာ Bastion Host လိုမလဲပေါ့ ။

အဲ့တာလေးမပြောခင် **Bastion Host** ဘာလဲဆိုတော့ Public Subnet မှာ ရှိတဲ့ EC2 instance ပါပဲ သူကဘာလုပ်ပေးလဲဆိုတော့ AWS VPC private subnet မှာရှိနေတဲ့ ကိုယ့်ရဲ့ အရေးကြီး server တွေကို SSH or RDP လှမ်းဝင်ပြီး manage လုပ်ဖို့.ပါပဲ။ သာမန်အားဖြင့် private subnet က instance တွေက Public IP မရှိပါဘူး ထို.ကြောင့် သူတို.ကို ကျွန်တော် တို. laptop ကနေ လှမ်း manage လုပ်ဖို. bastion host ကနေတဆင့်မှ ဝင်ရောက်ပြုလုပ်ရပါတယ်။

Bastion Host ဆိုတာ AWS ရဲ့ service တစ်ခုလားလို.မေးလာရင် မဟုတ်ပါဘူး၊ အမှန်က သူသည် အဓိကအားဖြင့် Linux Instance တစ်လုံးဖြစ်ပြီး SSH protocol ကို သုံးပြီး jump လုပ်ကာနောက်တစ်လုံးကို Login ဝင်ပြီး manage လုပ်တာပါ။Wi ndows ဆိုလည်း ထိုနည်းအတူပဲ RDP နဲ့jump လုပ်ကာ နောက်တစ်လုံးကို manage လုပ်တာပါပဲ။ ကျွန်တော်က ဒီ tutorial လေးမှာတော့ Linux instance တွေနဲ့လုပ်ပြမှာပါ။

ဒီနေရာမှာ မေးစရာရှိတာက ကျွန်တော် AWS သုံးတာ bastion host မသုံးပဲ private subnet က instance ကိုဝင်မရဘူးလား? ဒီလိုဆိုရင်တော့ ကျွန်တော်ရေးထားတဲ့ ဒီ **tutorial** လေးနဲ့လည်း အသုံးပြုနိုင်ပါတယ်။တခြားသော cloud provider တွေမှာလည်းဆိုင်ရာဆိုင်ရာတွေရှိနေမှာပါ။

ဒါဆို bastion host နဲ့ private subnet က ec2 instance တစ်လုံးကိုဘယ်လိုဝင်ပြီး manage လုပ်မလဲဆိုတာကြည့်ကြည့်ရအောင်။အောက်ကပုံလေးကနေ bastion host ရဲ့ overview diagram ကိုလေ့လာနိုင်ပါတယ်။

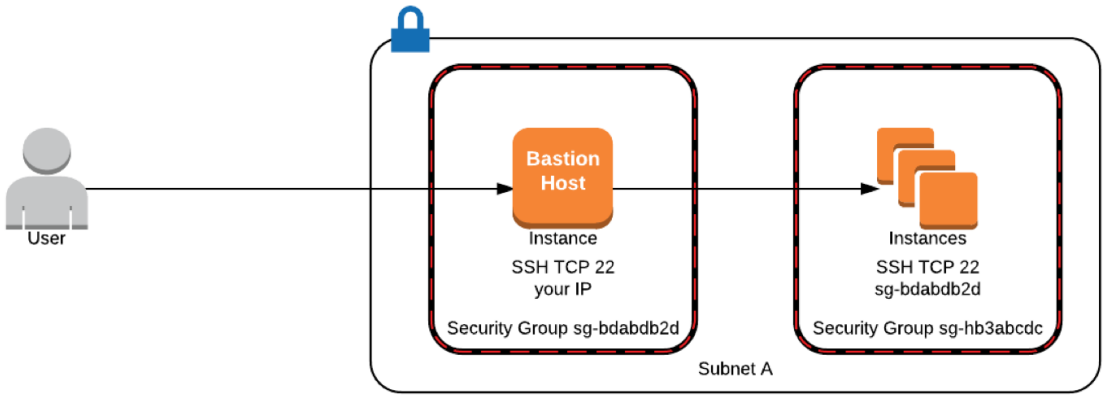


Photo reference - "aws.amazon.com"

ပုံမှာပြထားတဲ့အတင်းဆိုရင် bastion host ကို public subnet မှာထားပြီး user က private subnet က instance ကိုလှမ်း remote login ဝင်တာပါ။အဲ့ဒီမှာ ကျွန်တော်ပြောမှာက AWS မှာဆိုတော့ security group ကိုနည်းနည်း configure လုပ်ပေးရပါတယ်။အထူးသဖြင့် security တွက် ဆို ကျွန်တော် public subnet က bastion instance ရဲ့ security group မှာ ကျွန်တော်လက်ရှိ Public IP range ကိုပဲ SSH

Allow လုပ်ပါမယ် ပြီးနောက် private subnet က instance ရဲ့ security group မှာကတော့ bastion host ရဲ့ security group ကိုပဲ SSH Allow ပေးမှာပါ။Security အရဆိုရင်တော့ private subnet မှာရှိတဲ့ instance ကို bastion host ကနေပဲ access လုပ်လို့.ရမှာပါ။

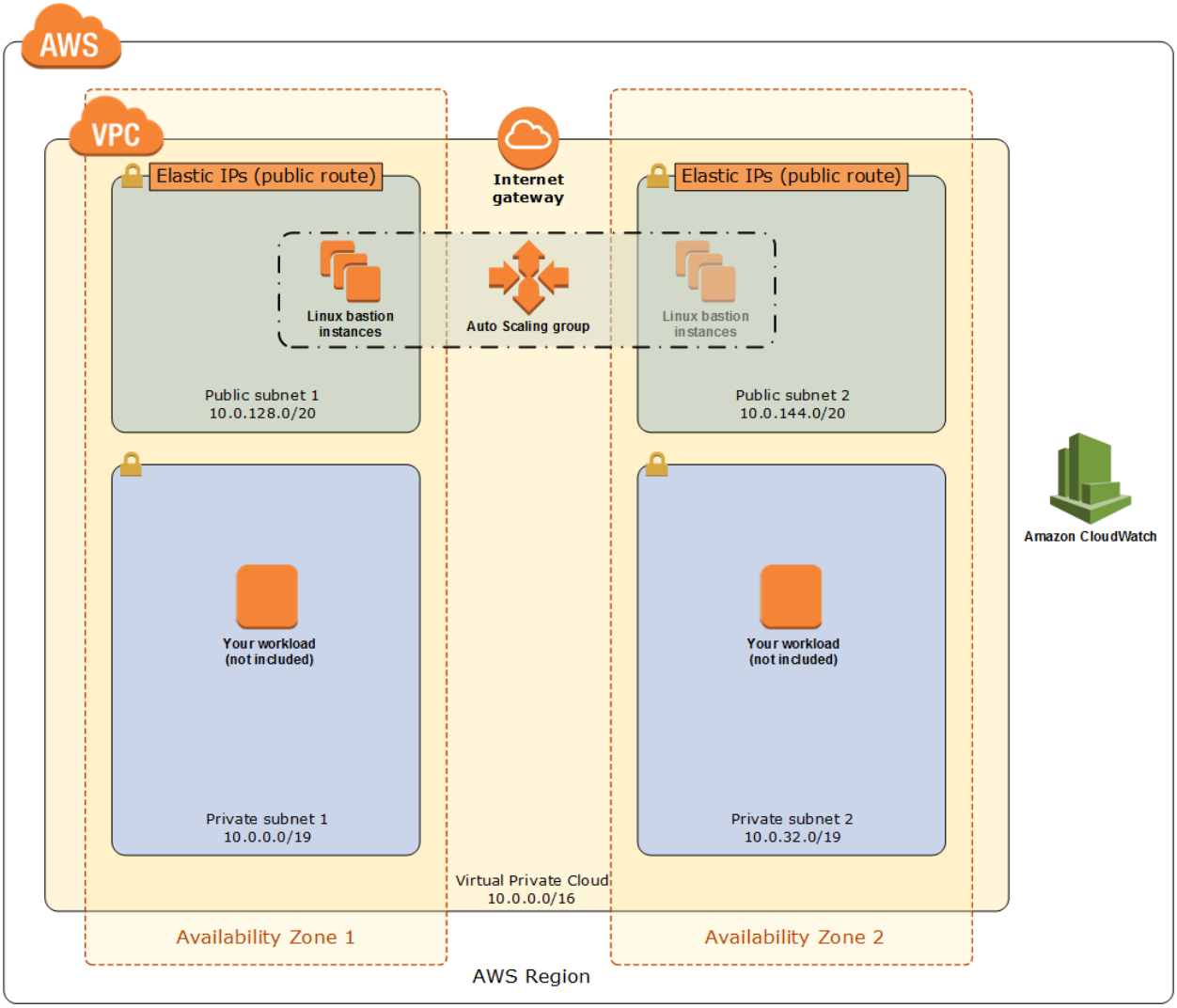
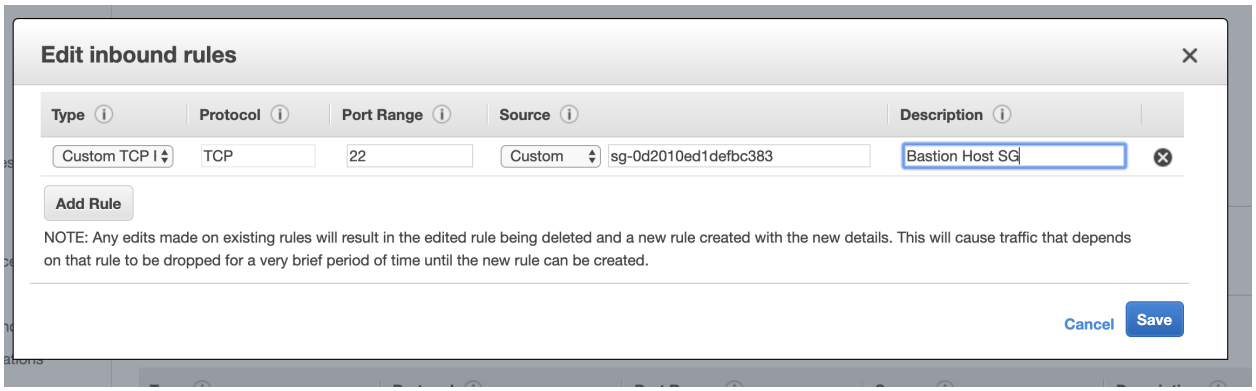


Photo reference - "aws.amazon.com"

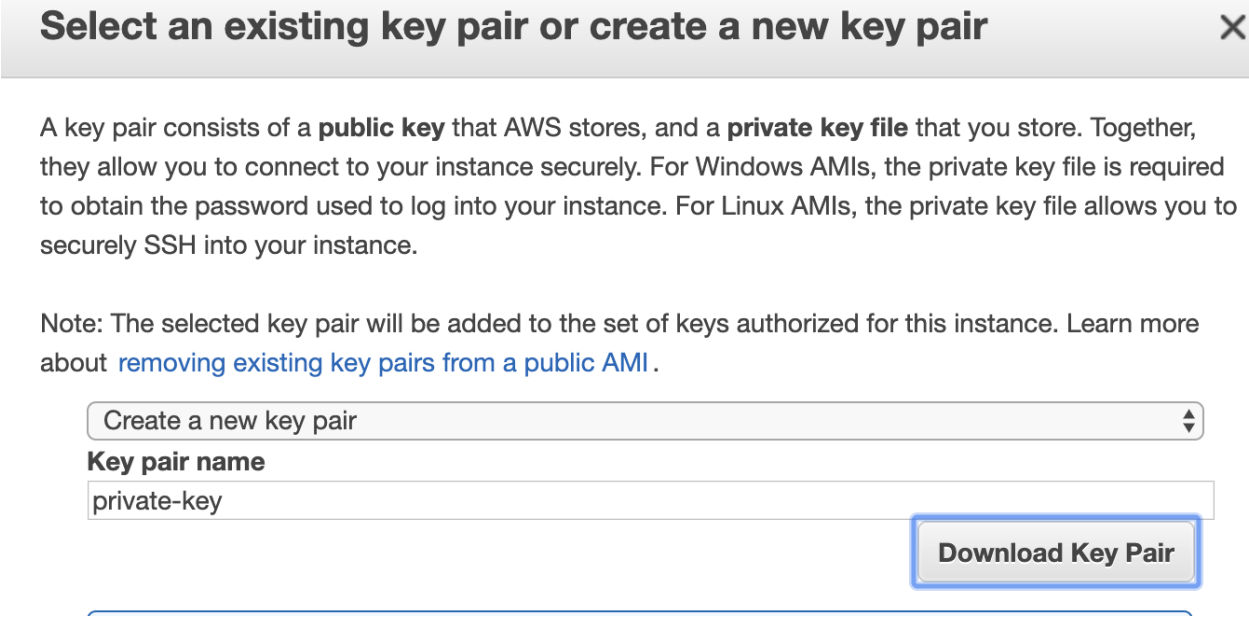
ဒါဆိုကျွန်တော်တို့. AWS management console ကို login ဝင်ပြီး EC2 တစ်လုံးကို private subnet မှာ ဆောက်လိုက်ပါမယ်။ Amazon Linux 2 Image နဲ့ပါ။

ယခုပြထားသော public domain များ IP များသည် Demonstration ပြရန်သာ ရည်ရွယ်ပါသည်။

ထိုနောက်ကျွန်တော်တို. private instance ရဲ့ security group မှာ ssh allow ကို bastion host ရဲ့ security group id ကိုပဲပေးထားပါမယ်။ဆိုလိုတာက ssh ကို bastion host ကနေပဲ access လုပ်လို.ရမှာပါ။



အားလုံး steps တွေပြီးသွားရင် PEM Key တစ်ခု create ပြီး instance ကို စတင် run လို.ရပါပြီ။



Pem Key Download ပြီးသွားတာနဲ့ Ec2 instance က private subnet မှာ running ဖြစ်နေတာကို မြင်တွေ.ရပါလိမ့်မယ်။

Name	Env	Owner	Project	Instance ID	Instance Type	Availability
private instance				i-0e64e38226bbe16…	t2.micro	us-east-1c

Instance: i-0e64e38226bbe1619 (private instance) **Private IP: 192.168.3.225**

Description | Status Checks | Monitoring | Tags

Instance ID	i-0e64e38226bbe1619	Public DNS (IPv4)	-
Instance state	running	IPv4 Public IP	-
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	
Private DNS	ip-192-168-3-225.ec2.internal	Availability zone	us-east-1c
Private IPs	192.168.3.225	Security groups	private instance security group. view inbound rules. view outbound rules

search : i-0e64e38226bbe1619 Add filter

Name	Instance ID	Instance Type	Availability Zone	Instance State
private instance	i-0e64e38226bbe16…	t2.micro	us-east-1c	running

ထိုနောက်ကျွန်တော်တို. bastion host ကို public subnet မှာဆောက်ပါမယ်။ Amazon Linux 2 Image နဲ့ပါပဲ။

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status
Bastion Host	i-001c76f411ada0ddb	t2.micro	us-east-1a	running	2/2 checks …	None

Instance: i-001c76f411ada0ddb (Bastion Host) **Public DNS: ec2-3-208-93-134.compute-1.amazonaws.com**

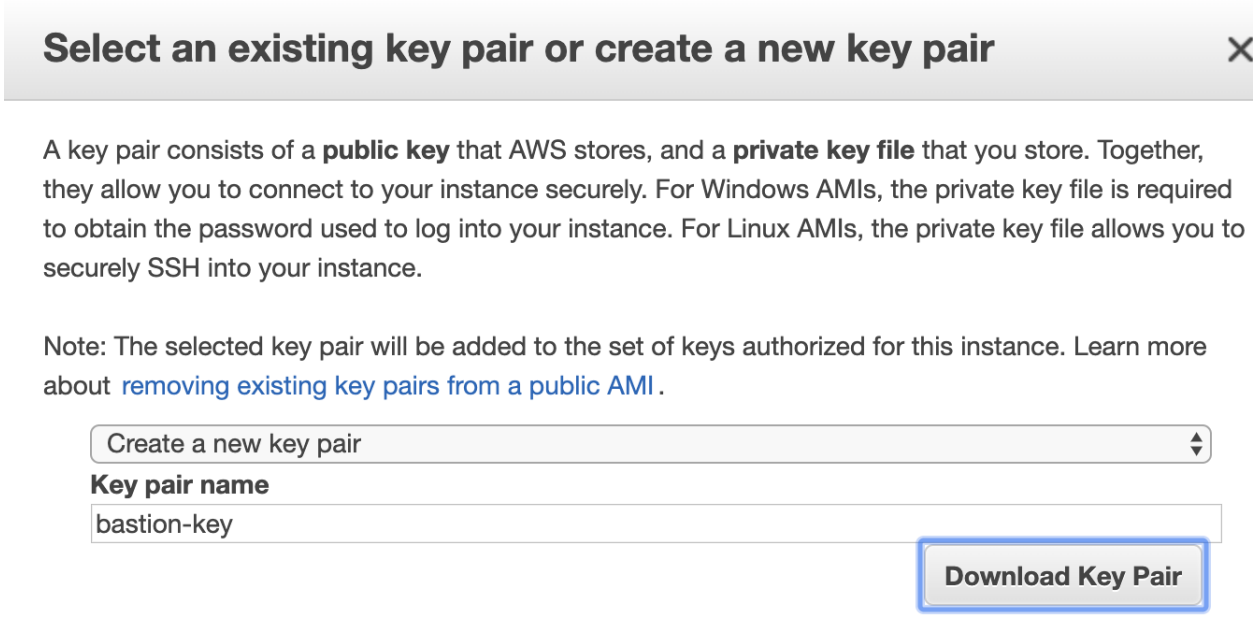
Description | Status Checks | Monitoring | Tags

Instance ID	i-001c76f411ada0ddb	Public DNS (IPv4)	ec2-3-208-93-134.compute-1.amazonaws.com
Instance state	running	IPv4 Public IP	3.208.93.134
Instance type	t2.micro	IPv6 IPs	-
Finding	Opt-in to AWS Compute Optimizer for recommendations. Learn more	Elastic IPs	

Name	Instance ID	Instance Type	Availability Zone	Instance State
Bastion Host	i-001c76f411ada0ddb	t2.micro	us-east-1a	running

ပြီးရင် private instance လိုပဲ security group မှာပြင်ပါမယ်။ Bastion host ရဲ့ security group မှာ ကျွန်တော် laptop ရဲ့ Public IP ကိုပဲ specific allow ပေးပါမယ်။Bastion host ကို ကျွန်တော်တစ်ယောက်တည်းကပဲ SSH access ရအောင်လို.ပါ။

ထုံးစံအတိုင်းနောက်ဆုံးမှာ PEM Key ကို Download ချပါမယ်။



နောက်ဆုံး အောက်ကပုံမှာပြထားတဲ့အတိုင်း EC2 နှစ်လုံး running ဖြစ်နေပါလိမ့်မယ်။တစ်ခုက public subnet နောက်တစ်ခုက private subnet မှာ run နေပါလိမ့်မယ်။

Name	Instance ID	Instance Type	Availability Zone	Instance State
Bastion Host	i-001c76f411ada0ddb	t2.micro	us-east-1a	running
private instance	i-0e64e38226bbe16…	t2.micro	us-east-1c	running

အားလုံး ဆောက်ပြီးပါက ကျွန်တော်တို.စပြီး bastion host ကနေတဆင့် private subnet က instance ကို login ဝင်ကြည့်ရအောင်။ မဝင်ခင်ကျွန်တော်တို. IP တွေ ssh username တွေတစ်ချက်ကြည့်ကြည့်ရအောင်။

Bastion Host Public URL = ec2-3-208-93-134.compute-1.amazonaws.com Private Instance IP = 192.168.3.225

ယခုပြထားသော public domain များ IP များသည် Demonstration ပြရန်သက်သက်သာရည်ရွယ်ပါသည်။

အိုကေ ဒါဆို ကျွန်တော်တို. local linux terminal ကနေ ssh proxy command လေးရိုက်ပြီး private instance ကိုလှမ်းဝင်ကြည့်ရအောင်။

$ ssh -i private-key.pem ec2-user@192.168.3.225 -o ProxyCommand="ssh -W %h:%p -i bastion-key.pem ec2-user@ec2-3-208-93-134.compute-1.amazonaws.com"

အထက်ပါ command လေးနဲ့ local ကနေ AWS private subnet မှာ ရှိတဲ့ instance လေးကို တိုက်ရိုက်တန်းဝင်လိုက်ပါတယ်။

$ ssh -i private-key.pem ec2-user@192.168.3.225 -o ProxyCommand="ssh -W %h:%p -i bastion-key.pem ec2-user@ec2-3-208-93-134.compute-1.amazonaws.com"

Last login: Thu Jan 9 17:49:38 2020 from ip-192-168-1-248.ec2.internal

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|
```

https://aws.amazon.com/amazon-linux-2/

[ec2-user@ip-192-168-3-225 ~]$ ip a | grep eth0

2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000

  inet 192.168.3.225/24 brd 192.168.3.255 scope global dynamic eth0

[ec2-user@ip-192-168-3-225 ~]$

ဒါဆိုကျွန်တော်တို. private instance ကိုဝင်သွားပါပြီ။ ဆိုတော့နောက်ဆုံး အနေနဲ့ ကျွန်တော်တို.အဲ့လို command အရှည်ကြီးကို မရိုက်ချင် မမှတ်ချင်ပါဘူး။ ဒါကြောင့်ကျွန်တော်တို. ssh config

လေးရေးပြီးအလွယ်တကူဝင်ရောက်ကြည့်ပါမယ်။ဒါဆိုကျနော်တို့. ssh config လေးကို **~/.ssh** ဆိုတဲ့ folder လေးမှာ **config** ဆိုပြီး ဆောက်ပြီး အောက်ပါအတိုင်း ဖြည့်လိုက်ပါမယ်။

```
$ vim ~/.ssh/config

Host private-instance

 Hostname 192.168.3.225

 IdentityFile  /Users/phyominhtun/Desktop/private-key.pem

 ForwardAgent yes

 User ec2-user

 ProxyCommand ssh -W %h:%p -i /Users/phyominhtun/Desktop/bastion-key.pem
ec2-user@ec2-3-208-93-134.compute-1.amazonaws.com
```

ယခုပြထားသော public domain များ IP များသည် Demonstration ပြရန်သက်သက်သာရည်ရွယ်ပါသည်။

ဒါဆို file လည်းဆောက်ပြီးပီ config လေးလည်းဖြည့်ပြီးပြီဆို ကျွန်တော်တို့. terminal ကနေအောက်ပါ command လေးရိုက်ပြီး private instance ကိုတိုက်ရိုက် Login လုပ်လိုရပါပြီ။

```
$ ssh private-instance

Last login: Thu Jan  9 17:56:04 2020 from ip-192-168-1-248.ec2.internal


       __|  __|_  )

       _|  (     /   Amazon Linux 2 AMI

      ___|\___|___|


https://aws.amazon.com/amazon-linux-2/

[ec2-user@ip-192-168-3-225 ~]$
```

ကျွန်တော်တို့. local laptop ကနေ AWS private subnet က instance ကို bastion host ကနေတဆင့်တိုက်ရိုက်ဝင်ပြီး manage လုပ်လို.ရပါပြီ။ နောက်ဆုံးအနေနဲ့ သိရှိရမည့်တစ်ချက်က bastion host က ကိုယ့် cloud network ကို အပြင်ကဝင်ပြီး private subnet က instance တွေကို

manage လုပ်ဖို့.အတွက်တာရည်ရွယ်ပြီး private subnet က instance တွေ internet စီပြန်လည်ထွက်ရှိဖို့.ကတော့ AWS မှာ NAT gateway or Nat instance ဆိုတာကို public subnet မှာထားပေးဖို့.လိုပါတယ်။ ဆိုတော့ ကျွန်တော် NAT gateway and NAT instance ဆိုတာလေးကို နောက်ပိုင်း content များတွင်ဖော်ပြပေးသွားမည်ဟုပြောကြားရင်း ဒီစာစုလေးကို အဆုံးသတ်ပါရစေ။

အားလုံးကိုကျေးဇူးတင်ပါသည်။ AWS User Group Myanmar

How to complete setup Kubernetes Cluster on AWS using Kops

Kops on AWS

Thinking Basic requirement

- Installing Cluster (IAC)
- Authentication (IAM)
- Authorization (K8s resources)
- Ingress Controller (ALB)
- External DNS

Prerequisites before setup cluster

- Kops
- aws cli
- aws-iam-authenticator
- kubectl

What is Kops ?

Kops သည် K for kubernetes၊ ops for Operations ၂ခု ပေါင်းသည့်အခါ Kubernetes Operations ဟုတ် အဓိပ္ပါယ်ရသည်။ အမျိုးမျိုးသော cloud platforms များပေါ်တွင် kubernetes cluster ကို version control system ဖြင့် install ပြုလုပ်ခြင်း၊ manage ပြုလုပ်ခြင်း၊ စသည့် operation များကို ကူညီ အထောက်အပံ့ ဖြစ်ရန် ရည်ရွယ် ထုတ်လုပ်ထားသည့် CLI tool တစ်ခုလည်း ဖြစ်ပါသည်။ Kops သည် kubernetes project အတူဖြစ်ပေါ်လာပြီး K8s team မှ ထိန်းသိမ်းသည့် tools တစ်ခု ဖြစ်သည်။

What is AWS CLI ?

AWS CLI သည် Developer များ၊ Ops များ အတွက် AWS cloud platform ၏ resources များကို command line မှ တဆင့် အလွယ်တကူ အသုံးပြုနိုင်ရန် အထူးပြုလုပ်ထားသော Amazon Web Services မှ tools တစ်ခုဖြစ်သည်။

What is aws-iam-authenticator ?

Kubernetes Cluster ကို AWS ၏ **IAM** ကို အသုံးပြု၍ authentication ပြုလုပ်နိုင်ရန် အတွက် ထုတ်လုပ်ထားသော tools တစ်ခုဖြစ်သည်။ aws-iam-authenticator သည် kubernetes special interest groups (SIG) ၏ project တစ်ခု ဖြစ်ပြီး ယခု လက်ရှိတွင် Heptio and Amazon EKS OSS မှ ထိန်းသိမ်ထားပါသည်။

What is kubectl ?

Kubectl သည် kubernetes cluster အတွက် client command line interface tool တစ်ခု ဖြစ်သည်။ kubectl သည် kubernetes cluster ကို ဆက်သွယ် ချိတ်ဆက် ရန် $HOME/.kube directory အောက် တွင် ရှိသောက် config file များကို ဖတ်သည်။ environment variable ***KUBECONFIG*** ကိုလဲ config file အစား အသုံး ပြုနိုင်သည်။

Let's SETUP

Kops ကို ဒီ link တွင် download ပြုလုပ်နိုင်ပါသည်။

Terminal

```
$ wget https://github.com/kubernetes/kops/releases/download/1.14.0-alpha.2/kops-linux-amd64

$ chmod +x kops-linux-amd64

$ sudo mv kops-linux-amd64 /usr/local/bin/kops
```

Kops ကို အသုံးပြုရန် AWS ၏ IAM user တစ်ယောက် လိုအပ်ပါသည်။ ထို IAM user တွင် Kops မှ AWS ၏ resource များကို အသုံးပြုရန် အနည်းဆုံး အောက်ပါ permission များ လိုအပ်ပါသည်။

- AmazonEC2FullAccess
- AmazonRoute53FullAccess
- AmazonS3FullAccess
- IAMFullAccess
- AmazonVPCFullAccess

လိုအပ်သော Permission နှင့် User တစ်ယောက်ဆောက်ပြီးသည့်အခါတွင် အောက်ဖော်ပြပါ Command နဲ့ AWS CLI ကို Setup လုပ်ပေးရမည်။

Terminal

```
$ aws configure
```

AWS CLI ကို setup လုပ်ပြီသောအခါ AWS ၏ S3 bucket တစ်ကို kops state file များ သိမ်းရန် အောက်ပါ command များကို အသုံးပြု၍ တည်ဆောက်နိုင်ပါသည်။

Terminal

```
$ export REGION=<your desire region>

$ export NAME=<your desire cluster name>

$ export KOPS_STATE_STORE=s3://${NAME} #your s3 bucket name

$ aws s3 mb $KOPS_STATE_STORE # create s3 bucket with your cluster name

$ aws s3api put-bucket-versioning --bucket ${NAME}  --versioning-configuration Status=Enabled

$ aws s3api put-bucket-encryption --bucket ${NAME} --server-side-encryption-configuration '{"Rules":[{"ApplyServerSideEncryptionByDefault":{"SSEAlgorithm":"AES256"}}]}'
```

AWS S3 တည်ဆောက်ပြီသောအခါ မာကျွန်တော်တို့ လိုချင်သည့် ပုံစံဖြင့် kubernetes cluster ကို တည်ဆောက်ရန် အဆင်သင့် ဖြစ်နေပြီပဲ ဖြစ်ပါသည်။ Kops ကို အသုံးပြု၍ private နဲ့ public topology ၂ မျိုးဖြင့် cluster ကို တည်ဆောက် နိုင်ပါသည်။

ဒီ scenario မှာတော့ private topology ကျွန်တော် အသုံးပြုသွားမှာ ဖြစ်သည်။ t3.medium instance type နှင့် weave network plugin ကို အသုံးပြုပြီး masters ၃ လုံး၊ nodes ၃လုံးကို အသုံးပြု၍တည်ဆောက်သွားမည်ဖြစ်ပါသည်။

Kops supports the following topologies on AWS

Topology	Value	Description
Public Cluster	public	All masters/nodes will be launched in a **public subnet** in the VPC
Private Cluster	private	All masters/nodes will be launched in a **private subnet** in the VPC

ထို.နောက် အောက်ဖော်ပြပါ Command နဲ့ လိုအပ်သော setup များဖြင့် cluster ကိုတည်ဆောက်ပြီး၊ထို cluster ရဲ့ config များကို ပြန်အသုံးပြုနိုင်ရန် **mycluster.yml** ဖိုင် ထဲကို standard output ဖြင့်ထည့်လိုက်ပါမယ်။

Terminal

```
$ kops create cluster ${NAME} --state=${KOPS_STATE_STORE} --master-count 3
--master-size t3.medium --node-count 3 --node-size t3.medium --zones
us-east-1a,us-east-1b,us-east-1c --topology private --networking weave --ssh-public-key
~/.ssh/id_rsa.pub  -o yaml >> mycluster.yaml
```

ထို.နောက် **mycluster.yml** ဖိုင်ကို edit လုပ်ပီး additional policies section တွင် AWS IAM Policy မှ ELB and Route 53 policy များကိုပြင်ဆင်ပြီးနောက် Cluster Authentication အတွက် Line number ၁၀ နှင့် ၁၁ တွင်ရှိသော အောက်ဖော်ပြပါ **authentication:aws:{}** ကိုထည့်ပေးရန်လိုအပ်ပါသည်။

AWS IAM နှင့် Authentication ကို Integrate မလုပ်ချင်ပါက ယခု Config ကို ပြင်ဆင်စရာမလိုအပ်ပါ။ပြီးတော့ aws-iam-authenticator ကို ထည့်သွင်းပေးရန်မလိုပါ။

mycluster.yaml

```
apiVersion: kops/v1alpha2

kind: Cluster

metadata:

 creationTimestamp: null

 name: <your cluster name> # no need to touch

spec:

 api:

   loadBalancer:

     type: Public

 authentication:      #need to add this [authenication]

   aws: {}            #need to add this [aws: {}]

 authorization:

   rbac: {}
```

```
channel: stable
cloudProvider: aws
configBase: s3://<your s3 name>/<your cluster name> #no need to touch
etcdClusters:
- cpuRequest: 200m
  etcdMembers:
  - instanceGroup: master-ap-southeast-1a
    name: a
  - instanceGroup: master-ap-southeast-1b
    name: b
  - instanceGroup: master-ap-southeast-1c
    name: c
  memoryRequest: 100Mi
  name: main
- cpuRequest: 100m
  etcdMembers:
  - instanceGroup: master-ap-southeast-1a
    name: a
  - instanceGroup: master-ap-southeast-1b
    name: b
  - instanceGroup: master-ap-southeast-1c
    name: c
  memoryRequest: 100Mi
  name: events
```

```
iam:
  allowContainerRegistry: true
  legacy: false
kubelet:
  anonymousAuth: false
kubernetesApiAccess:
- 0.0.0.0/0
kubernetesVersion: 1.14.1
masterPublicName: api.yourcluster.com  #no need to touch
networkCIDR: 172.20.0.0/16
networking:
  weave:
    mtu: 8912
nonMasqueradeCIDR: 100.64.0.0/10
additionalPolicies:     #add this additional policy from this line to until line 160
  node: |
    [
      {
        "Effect": "Allow",
        "Action": [
          "acm:DescribeCertificate",
          "acm:ListCertificates",
          "acm:GetCertificate"
        ],
```

```
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateSecurityGroup",
    "ec2:CreateTags",
    "ec2:DeleteTags",
    "ec2:DeleteSecurityGroup",
    "ec2:DescribeInstances",
    "ec2:DescribeInstanceStatus",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeTags",
    "ec2:DescribeVpcs",
    "ec2:ModifyInstanceAttribute",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:RevokeSecurityGroupIngress"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
```

```
"Action": [
  "elasticloadbalancing:AddTags",
  "elasticloadbalancing:CreateListener",
  "elasticloadbalancing:CreateLoadBalancer",
  "elasticloadbalancing:CreateRule",
  "elasticloadbalancing:CreateTargetGroup",
  "elasticloadbalancing:DeleteListener",
  "elasticloadbalancing:DeleteLoadBalancer",
  "elasticloadbalancing:DeleteRule",
  "elasticloadbalancing:DeleteTargetGroup",
  "elasticloadbalancing:DeregisterTargets",
  "elasticloadbalancing:DescribeListeners",
  "elasticloadbalancing:DescribeLoadBalancers",
  "elasticloadbalancing:DescribeLoadBalancerAttributes",
  "elasticloadbalancing:DescribeRules",
  "elasticloadbalancing:DescribeSSLPolicies",
  "elasticloadbalancing:DescribeTags",
  "elasticloadbalancing:DescribeTargetGroups",
  "elasticloadbalancing:DescribeTargetGroupAttributes",
  "elasticloadbalancing:DescribeTargetHealth",
  "elasticloadbalancing:ModifyListener",
  "elasticloadbalancing:ModifyLoadBalancerAttributes",
  "elasticloadbalancing:ModifyRule",
  "elasticloadbalancing:ModifyTargetGroup",
```

"elasticloadbalancing:ModifyTargetGroupAttributes",
"elasticloadbalancing:RegisterTargets",
"elasticloadbalancing:RemoveTags",
"elasticloadbalancing:SetIpAddressType",
"elasticloadbalancing:SetSecurityGroups",
"elasticloadbalancing:SetSubnets",
"elasticloadbalancing:SetWebACL"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"iam:GetServerCertificate",
"iam:ListServerCertificates"
],
"Resource": "*"
},
{
"Effect": "Allow",
"Action": [
"waf-regional:GetWebACLForResource",
"waf-regional:GetWebACL",
"waf-regional:AssociateWebACL",

```
    "waf-regional:DisassociateWebACL"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "tag:GetResources",
    "tag:TagResources"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "waf:GetWebACL"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "route53:*"
  ],
```

"Resource": "*"
}
]
sshAccess:
- 0.0.0.0/0
subnets:
- cidr: 172.20.32.0/19
  name: ap-southeast-1a
  type: Private
  zone: ap-southeast-1a
- cidr: 172.20.64.0/19
  name: ap-southeast-1b
  type: Private
  zone: ap-southeast-1b
- cidr: 172.20.96.0/19
  name: ap-southeast-1c
  type: Private
  zone: ap-southeast-1c
- cidr: 172.20.0.0/22
  name: utility-ap-southeast-1a
  type: Utility
  zone: ap-southeast-1a
- cidr: 172.20.4.0/22
  name: utility-ap-southeast-1b

```
    type: Utility
    zone: ap-southeast-1b
  - cidr: 172.20.8.0/22
    name: utility-ap-southeast-1c
    type: Utility
    zone: ap-southeast-1c
  topology:
    dns:
      type: Public
    masters: private
    nodes: private

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: null
  labels:
    kops.k8s.io/cluster: hearu.org
  name: master-ap-southeast-1a
spec:
  image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
  machineType: t3.medium
```

```
 maxSize: 1
 minSize: 1
 nodeLabels:
   kops.k8s.io/instancegroup: master-ap-southeast-1a
 role: Master
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1a

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com # no need to touch
 name: master-ap-southeast-1b
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 1
 minSize: 1
```

```
 nodeLabels:
   kops.k8s.io/instancegroup: master-ap-southeast-1b
 role: Master
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1b

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com #no need to touch
 name: master-ap-southeast-1c
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 1
 minSize: 1
 nodeLabels:
   kops.k8s.io/instancegroup: master-ap-southeast-1c
```

```
 role: Master
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1c

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com #no need to touch
 name: nodes
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 3
 minSize: 3
 nodeLabels:
   kops.k8s.io/instancegroup: nodes
 role: Node
 rootVolumeSize: 60
```

```
 rootVolumeType: gp2

 subnets:

 - ap-southeast-1a

 - ap-southeast-1b

 - ap-southeast-1c
```

အားလုံးပြင်ဆင်ပြီးနောက် cluster ကို အောက်ပါ command များဖြင့် bootstrap လုပ်နိုင်ပါပြီ။ Bootstrap လုပ်နေချိန်တွင် မိနစ်အနည်းငယ် စောင့်ရန်လိုအပ်ပါသည်။

Terminal

```
$ kops create -f mycluster.yaml

$ kops create secret --name ${NAME} sshpublickey admin -i ~/.ssh/id_rsa.pub

$ kops update cluster ${NAME} --yes

$ kops kops validate cluster # validate cluster is check for your cluster state
```

Cluster တည်ဆောက်ပြီးဖြစ်သော်လည်း Status ကတော့ valid ဖြစ်သေးမှာမဟုတ်ပါဘူး။ ဘာလို့လဲဆိုတော့ authentication plugin ၏ configuration လိုနေသောကြောင့်ဖြစ်ပါသည်။

json

```
//K8sAdmin

{

   "Version": "2012-10-17",

   "Statement": [

       {

           "Sid": "VisualEditor0",

           "Effect": "Allow",

           "Action": [
```

```
            "sts:DecodeAuthorizationMessage",
            "sts:GetCallerIdentity"
        ],
        "Resource": "*"
    },
    {
        "Sid": "VisualEditor1",
        "Effect": "Allow",
        "Action": "sts:*",
        "Resource": "arn:aws:iam::<account id>:role/K8sAdmin"
    }
  ]
}
```

json

//K8sDeveloper

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Sid": "VisualEditor0",
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::<account id:role/K8sDeveloper"
    },
```

```
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
            "Resource": "*"
        }
    ]
}
```

အောက်တွင်ဖော်ပြထားသော commands ဖြင့် K8sDeveloper နှင့် K8sAdmin အစရှိသော role နှစ်ခုကို configmap အတွက် တည်ဆောက်သည့်အခါတွင် အပေါ်မှာ json များထွက်လာမည်ဖြစ်ပါသည်။

Example ပြထားခြင်းဖြစ်ပါသည်။

Terminal

```
$ export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query 'Account')

$ export DEVPOLICY=$(echo -n
'{"Version":"2012-10-17","Statement":[{"Sid":"VisualEditor0","Effect":"Allow","Action":["sts:Decod
eAuthorizationMessage","sts:GetCallerIdentity"],"Resource":"*"},{"Sid":"VisualEditor1","Effect":"A
llow","Action":"sts:*","Resource":"arn:aws:iam::'; echo -n "$ACCOUNT_ID"; echo -n
':role/K8sDeveloper"}]}')

$ aws iam create-role \
 --role-name K8sDeveloper \
 --description "Kubernetes developer role (for AWS IAM Authenticator for AWS)." \
 --assume-role-policy-document "$DEVPOLICY" \
 --output text \
 --query 'Role.Arn'
```

```
$ export ADMPOLICY=$(echo -n
'{"Version":"2012-10-17","Statement":[{"Sid":"VisualEditor0","Effect":"Allow","Action":["sts:Decod
eAuthorizationMessage","sts:GetCallerIdentity"],"Resource":"*"},{"Sid":"VisualEditor1","Effect":"A
llow","Action":"sts:*","Resource":"arn:aws:iam::'; echo -n "$ACCOUNT_ID"; echo -n
':role/K8sAdmin"}]}')

$ aws iam create-role \

 --role-name K8sAdmin \

 --description "Kubernetes administrator role (for AWS IAM Authenticator for AWS)." \

 --assume-role-policy-document "$ADMPOLICY" \

 --output text \

 --query 'Role.Arn'
```

ကျွန်တော်တို. IAM role နှင့် policies များကိုတည်ဆောက်ပြီးနောက် Kubernets resources များတွက် authorization ကို configure ချပေးရန်လိုအပ်ပါသည်။

ထို.နောက် RBAC authorization ကို အသုံးပြုပြီး developer role အတွက် production နှင့် development namespace နှစ်ခုကို configure ချပေးထားပါသည်။

K8sDeveloper role အတွင်းရှိသော user များသည် ထို namespace နှစ်ခုနှင့် resource အချို.သာလျင် အသုံးပြုနိုင်မည်ဖြစ်သည်။

developer-rbac.yaml

```
# RBAC Policy for developer

# create 2 Roles on for developer and binding roles

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: Role

metadata:

 namespace: development

 name: developer

rules:

- apiGroups: ["*"]
```

```
 resources:
["deployments","configmaps","pods","secrets","services","endpoints","persistentvolumeclaims","
events","namespaces","statefulsets","replicasets","jobs","cronjobs"]

 verbs: ["create","get","delete","list","update","edit","watch","exec","patch"]

- apiGroups:

 - '*'

 resources:

 - 'pods/exec'

 - 'pods/portforward'

 - 'services/portforward'

 verbs:

 - create

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

 name: developer

 namespace: development

roleRef:

 kind: Role

 name: developer

 apiGroup: rbac.authorization.k8s.io

subjects:

- kind: Group
```

```
 namespace: development

 name: k8s.developer

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: Role

metadata:

 namespace: production

 name: developer

rules:

- apiGroups: ["*"]

 resources:
["deployments","configmaps","pods","secrets","services","endpoints","persistentvolumeclaims","
events","namespaces","statefulsets","replicasets","jobs","cronjobs"]

 verbs: ["create","get","delete","list","update","edit","watch","exec","patch"]

- apiGroups:

 - '*'

 resources:

 - 'pods/exec'

 - 'pods/portforward'

 - 'services/portforward'

 verbs:

 - create

---

apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: RoleBinding

metadata:

 name: developer

 namespace: production

roleRef:

 kind: Role

 name: developer

 apiGroup: rbac.authorization.k8s.io

subjects:

- kind: Group

 namespace: production

 name: k8s.developer
```

Terminal

```
$ kubectl create namespace development

$ kubectl create namespace production

$ kubectl create -f developer-rbac.yaml
```

aws-iam-authenticator အလုပ်လုပ်ရန် နှင့် သတ်မှတ်ထားသည့် user group policy များ အသက်ဝင်ရန် အောက်ပါ configmap ကို create ပြုလုပ်ပေးရန်လိုအပ်ပါသည်။

aws-auth.yaml

```
apiVersion: v1

kind: ConfigMap

metadata:

 namespace: kube-system

 name: aws-iam-authenticator
```

```
 labels:
   k8s-app: aws-iam-authenticator
data:
 config.yaml: |
   clusterID: api.mycluster.com
   server:
     mapRoles:
     - roleARN: arn:aws:iam::<account id>:role/K8sAdmin
       username: kubernetes-admin
       groups:
       - system:masters
     - roleARN: arn:aws:iam::<account id>:role/K8sDeveloper
       username: kubernetes-developer
       groups:
       - k8s.developer
```

Terminal

```
$ kubectl create -f aws-auth.yaml
```

ယခုအခြေအနေတွင် Kubernetes cluster နှင့် authentication/authorization setup ပြုလုပ်ခြင်းမှာ ပြီးမြောက်လုနီးပါးဖြစ်ပါသည်။ Kubernetes client (kubectl) မှ cluster ကို access လုပ်ရန် configure ချဖို. လိုအပ်ပါသည်။ အောက်ဖော်ပြပါ config နှစ်ခုသည် Developer နှင့် Administrator အတွက် ဖြစ်ပါသည်။ ကျွန်တော် ကတော့ Admin config အသုံးပြုပြီး AWS Application LoadBalancer ကိုဆက်လက် setup လုပ်သွားမှာ ဖြစ်ပါသည်။

```
~/kube/.config (Admin)

# this is for Kubernetes Administrator

# you must change this the value of 'certificate-authority-data' with your ca.
```

apiVersion: v1

clusters:

- cluster:

  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMwekNDQWJ1Z0F3SUJBZ0lNRmFlcH
Bua0lEUnRwNkpub01BMEdDU3FHU0liM0RRRUJDd1VBTUJVeEV6QVIKQmdOVkJBTVRDbX
QxWW1WeWJtVjBaWE13SGhjTk1Ua3dOakV4TURVd05USTRXaGNOTWprd05qRXdNRFV3Tl
RJNApXakFWTVJNd0VRWURWUVFERXdwcmRXSmxjbTVsZEdWek1JSUJJakFOQmdrcWhra
Uc5dzBCQVFFRkFBT0NBUThBCk1JSUJDZ0tDQVFFQXpmMzlOOUIvaGpUQnViM0tETEhzM1
NJUmhVL2pVcUt2RWhTWnpCQngwR0tXNmtONmRUVkkKK3dwUnRoeU9Yd2ZWRHA4RmxR
YUFYbWt0UGtGTG9jU2VSelBUWXRod1VBamhSbVNoZFJtcW1yaXBZTlhFdEZYMAp6SkZFS
2NRNDJ0TEhYanZZbVc3SGxYc0FDSDRCKy82SEtwVmxGSVdNTDAwb2tTUmEzZVBvdTdW
YzlqTkZhd3R0CmFoMThrNStLcHVoL0FyUEs2dmdhcEhLclB2dUNaZG9FWkNtaE4rSVdZTjN6
Uk1Wb1A1TEVoTEx5RFFzUzYvN0sKQ3prQ09Pam1QWW01UmxnOVRvb25ZcStwbS9LMXc4
V1NLVnZick9VZXF0dXkxUlJYWUxtc0VNVmx0NDRQSS9YTgpEeUlubmtpVmRwc1A4ejAwRjZ
5MGEwYXFpT3RGUVVyd3NRSURBUUFCb3lNd0lUQU9CZ05WSFE4QkFmOEVCQU1DCkFR
WXdEd1lEVlIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFq
Um5ndnJ6SmRMWm8KRFdlbVFaMi83N0xLTmkzTmpRQU9NZUg3WXVpZnlNbTZNdTB2V3F6
UU54SDJ1TGhFd0JRN0VocEpYSnFoMmp0dQpldnR6YUF6N2o5MEVKdlN6WHRtSDRQaHd1
NFZFMHJWRGVlbUhrMXRZRjhsQ3hVM2xtZFRGZjJ4M29qVnVKNndpCkd3c3RaU0s5V2J5V
W53RjNLUXdkMndmd3VUVVpSRGtuSkI2ZGhOcUxNakN3V1M1ekpDUjcvYXFuM0tqb0xvWEg
KWkI1VW1UcGd2Sm9rWm0wcUF0ODRNS0hFb2ZCYXJ0M1BUcnplWGI0N1VHRlMrM0h1anV
QUTBreW1DT0crK1ZEMwo0K0M5eVlXaTRhYnd6SE9rTmQvMlFKZVoyQjhjbjhlVU5kUGFSSnk
weS9keVpSd0NZdXFKSkxZeTVGUTFiQlUwCm96bThnMG01ZWc9PQotLS0tLUVORCBDRVJ
USUZJQ0FURS0tLS0tCg==

  server: https://api.yourcluster.com

 name: yourcluster.com

contexts:

- context:

  cluster: yourcluster.com

  user: kubernetes-admin

 name: aws

current-context: aws

```
kind: Config

preferences: {}

users:

- name: kubernetes-admin

 user:

   exec:

     apiVersion: client.authentication.k8s.io/v1alpha1

     args:

     - token

     - -i

     - api.yourcluster.com

     - -r

     - arn:aws:iam::<account id>:role/K8sAdmin

     command: aws-iam-authenticator

     env: null
```

~/kube/.config (Developer)

```
# this is for Kubernetes Developer

# you must change this the value of 'certificate-authority-data' with your ca.

apiVersion: v1

clusters:

- cluster:

   certificate-authority-data:
```

LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMwekNDQWJ1Z0F3SUJBZ0lNRmFlcH
Bua0lEUnRwNkpub01BMEdDU3FHU0liM0RRRUJDd1VBTUJVeEV6QVIKQmdOVkJBTVRDbX
QxWW1WeWJtVjBaWE13SGhjTk1Ua3dOakV4TURVd05USTRXaGNOTWprd05qRXdNRFV3Tl
RJNApXakFWTVJNd0VRWURWUVFERXdwcmRXSmxjbTVsZEdWek1JSUJJakFOQmdrcWhra

Uc5dzBCQVFFRkFBT0NBUThBCk1JSUJDZ0tDQVFFQXpmMzlOOUIvaGpUQnViM0tETEhzM1
NJUmhVL2pVcUt2RWhTWnpCQngwR0tXNmtONmRUVkkKK3dwUnRoeU9Yd2ZWRHA4RmxR
YUFYbWt0UGtGTG9jU2VSelBUWXRod1VBamhSbVNoZFJtcW1yaXBZTlhFdEZYMAp6SkZFS
2NRNDJ0TEhYanZZbVc3SGxYc0FDSDRCKy82SEtwVmxGSVdNTDAwb2tTUmEzZVBvdTdW
YzlqTkZhd3R0CmFoMThrNStLcHVoL0FyUEs2dmdhcEhLclB2dUNaZG9FWkNtaE4rSVdZTjN6
Uk1Wb1A1TEVoTEx5RFFzUzYvN0sKQ3prQ09Pam1QWW01UmxnOVRvb25ZcStwbS9LMXc4
V1NLVnZick9VZXF0dXkxUlJYWUxtc0VNVmx0NDRQSS9YTgpEeUlubmtpVmRwc1A4ejAwRjZ
5MGEwYXFpT3RGUVVyd3NRSURBUUFCb3lNd0lUQU9CZ05WSFE4QkFmOEVCQU1DCkFR
WXdEd1lEVlIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFq
Um5ndnJ6SmRMWm8KRFdlbVFaMi83N0xLTmkzTmpRQU9NZUg3WXVpZnlNbTZNdTB2V3F6
UU54SDJ1TGhFd0JRN0VocEpYSnFoMmp0dQpldnR6YUF6N2o5MEVKdlN6WHRtSDRQaHd1
NFZFMHJWRGVlbUhrMXRZRjhsQ3hVM2xtZFRGZjJ4M29qVnVKNndpCkd3c3RaU0s5V2J5V
W53RjNLUXdkMndmd3VUVVpSRGtuSkI2ZGhOcUxNakN3V1M1ekpDUjcvYXFuM0tqb0xvWEg
KWkI1VW1UcGd2Sm9rWm0wcUF0ODRNS0hFb2ZCYXJ0M1BUcnplWGI0N1VHRlMrM0h1anV
QUTBreW1DT0crK1ZEMwo0K0M5eVlXaTRhYnd6SE9rTmQvMlFKZVoyQjhjbjhlVU5kUGFSSnk
weS9keVpSd0NZdXFKSkxZeTVGUTFiQlUwCm96bThnMG01ZWc9PQotLS0tLUVORCBDRVJ
USUZJQ0FURS0tLS0tCg==

  server: https://api.yourcluster.com

 name: yourcluster.com

contexts:

- context:

  cluster: yourcluster.com

  namespace: development #you will notiace this. This is means developere default namesapce is development.

  user: kubernetes-developer

 name: aws

current-context: aws

kind: Config

preferences: {}

users:

- name: kubernetes-developer

```
 user:
   exec:
     apiVersion: client.authentication.k8s.io/v1alpha1
     args:
     - token
     - -i
     - api.yourcluster.com
     - -r
     - arn:aws:iam::<account id>:role/K8sDeveloper
     command: aws-iam-authenticator
     env: null
```

Terminal

```
$ wget https://github.com/kubernetes-sigs/aws-iam-authenticator/releases/download/v0.4.0/aws-iam-authenticator_0.4.0_linux_amd64

$ chmod +x aws-iam-authenticator_0.4.0_linux_amd64

$ sudo mv aws-iam-authenticator_0.4.0_linux_amd64 /usr/local/bin/aws-iam-authenticator

$ aws-iam-authenicator version

{"Version":"0.4.0","Commit":"9999cd8047782c94c08e2d24b77dc9488947aa03"}

$ kubectl get nodes # test what you authorized
```

ထို့နောက် AWS Application LoadBalancer ကို Kubernetes Ingress Controller နဲ့ integrate ပြုလုပ်ရန် အောက်ဖော်ပြပါအတိုင်း RBAC နှင့် alb-ingress-controller deployment ကို kube-system တွင် create လုပ်ရမည်ဖြစ်ပါသည်။ လိုအပ်သော additional policy များကို cluster bootstrap လုပ်ခဲ့တုန်း ကထည့်သွင်းခဲ့ပြီးဖြစ်ပါသည်။

alb-rbac.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
kind: ClusterRole
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress-controller
rules:
 - apiGroups:
     - ""
     - extensions
   resources:
     - configmaps
     - endpoints
     - events
     - ingresses
     - ingresses/status
     - services
   verbs:
     - create
     - get
     - list
     - update
     - watch
     - patch
 - apiGroups:
```

```
    - ""
    - extensions
  resources:
    - nodes
    - pods
    - secrets
    - services
    - namespaces
  verbs:
    - get
    - list
    - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress-controller
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: alb-ingress-controller
subjects:
```

```
 - kind: ServiceAccount
   name: alb-ingress
   namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress
 namespace: kube-system
```

alb-ingress-controller.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress-controller
 namespace: kube-system
spec:
 replicas: 1
 selector:
   matchLabels:
     app: alb-ingress-controller
```

```
  strategy:
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 1
    type: RollingUpdate
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: alb-ingress-controller
    spec:
      containers:
        - args:
            - --ingress-class=alb
            - --cluster-name=yourcluster.com
          env:
          image:
894847497797.dkr.ecr.us-west-2.amazonaws.com/aws-alb-ingress-controller:v1.0.0
          imagePullPolicy: Always
          name: server
          resources: {}
          terminationMessagePath: /dev/termination-log
      dnsPolicy: ClusterFirst
      restartPolicy: Always
```

```
    securityContext: {}

    terminationGracePeriodSeconds: 30

    serviceAccountName: alb-ingress

    serviceAccount: alb-ingress
```

Terminal

```
# setup with your administrator account.

$ kubectl create -f alb-rbac.yaml

$ kubectl create -f alb-ingress-controller.yaml

$ kubectl get pod -n kube-system
```

Ingress controller setup လုပ်ပီးနောက် external DNS အတွက် DNS records များကို Ingress create လုပ်လိုက်သည်နှင့် တပြိုင်နက်ထဲ AWS route 53 တွင် automatically update ပြုလုပ်သွားရန် အောက်ဖော်ပြပါ service account, rbac for external dns, deployment များကို ဆက်လက်တည်ဆောက်ရမည်ဖြစ်သည်။

external-dns.yaml

```
apiVersion: v1

kind: ServiceAccount

metadata:

 name: external-dns

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRole

metadata:

 name: external-dns

rules:

- apiGroups: [""]
```

```
  resources: ["services"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["get","watch","list"]
- apiGroups: ["extensions"]
  resources: ["ingresses"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
- kind: ServiceAccount
  name: external-dns
  namespace: kube-system
```

```
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
 name: external-dns
spec:
 strategy:
   type: Recreate
 template:
   metadata:
     labels:
       app: external-dns
   spec:
     serviceAccountName: external-dns
     containers:
     - name: external-dns
       image: registry.opensource.zalan.do/teapot/external-dns:v0.5.9
       args:
       - --source=service
       - --source=ingress
       - --domain-filter=yourcluster.com # will make ExternalDNS see only the hosted zones matching provided domain, omit to process all available hosted zones
       - --provider=aws
```

- --policy=upsert-only # would prevent ExternalDNS from deleting any records, omit to enable full synchronization

- --aws-zone-type=public # only look at public hosted zones (valid values are public, private or no value for both)

- --registry=txt

- --txt-owner-id=my-identifier

Terminal

$ kubectl create -f external-dns.yaml

ယခုအခြေအနေတွင်အားလုံးတည်ဆောက်ပြီး ဖြစ်ပါသည်။

တည်ဆောက်ခဲ့သည့် Cluster, IAM authentication and k8s RBAC, ALB ingress and automatic external DNS service များကို စမ်းသပ်ရန်အတွက် အောက်ဖော်ပြပါ 2048 web application game ဖြင့်စမ်းသပ်နိုင်ပါသည်။

Terminal

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-namespace.yaml

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-deployment.yaml

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-service.yaml

$ kubectl get po,deploy,svc -n 2048-game

2048-ingress.yaml

apiVersion: extensions/v1beta1

kind: Ingress

metadata:

```
 name: "2048-ingress"
 namespace: "2048-game"
 annotations:
   kubernetes.io/ingress.class: alb
   alb.ingress.kubernetes.io/scheme: internet-facing
 labels:
   app: 2048-ingress
spec:
 rules:
 - host: yourcluster.com
   http:
     paths:
     - path: /*
       backend:
         serviceName: "service-2048"
         servicePort: 80
```

Terminal

```
$ kubectl create -f 2048-ingress.yaml
$ kubectl get ingress -n 2048-game
$ dig yourcluster.com # see and wait
$ google-chrome-stable yourcluster.com
```

ယခု ကျွန်တော်တို. လုပ်ပြထားသော tutorial လေးသည် Non-Profit Organization တစ်ခုအတွက် ကျွန်တော်တို. Setup လုပ်ပေးခဲ့သော Production Grade Kops Cluster Setup Use-case မှ ကောက်နုတ်ဖော်ပြထားခြင်းဖြစ်သည့်အတွက် စာဖတ်သူတို.စိတ်ထဲတွင်

မရှင်းမလင်းတစ်ခုခုခံစားခဲ့ရရင်ကျွန်တော်တို့.ရဲ့ Facebook Group or Page ကိုဆက်သွယ်မေးမြန်းနိုင်ပါသည်။

How to complete setup Kubernetes Cluster on AWS using Kops

Kops on AWS

Thinking Basic requirement

- Installing Cluster (IAC)
- Authentication (IAM)
- Authorization (K8s resources)
- Ingress Controller (ALB)
- External DNS

Prerequisites before setup cluster

- Kops
- aws cli
- aws-iam-authenticator
- kubectl

What is Kops ?

Kops သည် K for kubernetes၊ ops for Operations ၂ခု ပေါင်းသည့်အခါ Kubernetes Operations ဟုတ် အဓိပ္ပါယ်ရသည်။ အမျိုးမျိုးသော cloud platforms များပေါ်တွင် kubernetes cluster ကို version control system ဖြင့် install ပြုလုပ်ခြင်း၊ manage ပြုလုပ်ခြင်း၊ စသည့် operation များကို ကူညီ အထောက်အပံ့ ဖြစ်ရန် ရည်ရွယ် ထုတ်လုပ်ထားသည့် CLI tool တစ်ခုလည်း ဖြစ်ပါသည်။ Kops သည် kubernetes project အတူဖြစ်ပေါ်လာပြီး K8s team မှ ထိန်းသိမ်းသည့် tools တစ်ခု ဖြစ်သည်။

What is AWS CLI ?

AWS CLI သည် Developer များ၊ Ops များ အတွက် AWS cloud platform ၏ resources များကို command line မှ တဆင့် အလွယ်တကူ အသုံးပြုနိုင်ရန် အထူးပြုလုပ်ထားသော Amazon Web Services မှ tools တစ်ခုဖြစ်သည်။

What is aws-iam-authenticator ?

Kubernetes Cluster ကို AWS ၏ **IAM** ကို အသုံးပြု၍ authentication ပြုလုပ်နိုင်ရန် အတွက် ထုတ်လုပ်ထားသော tools တစ်ခုဖြစ်သည်။ aws-iam-authenticator သည် kubernetes special interest groups (SIG) ၏ project တစ်ခု ဖြစ်ပြီး ယခု လက်ရှိတွင် Heptio and Amazon EKS OSS မှ ထိန်းသိမ်ထားပါသည်။

What is kubectl ?

Kubectl သည် kubernetes cluster အတွက် client command line interface tool တစ်ခု ဖြစ်သည်။ kubectl သည် kubernetes cluster ကို ဆက်သွယ် ချိတ်ဆက် ရန် $HOME/.kube directory အောက် တွင် ရှိသောက် config file များကို ဖတ်သည်။ environment variable ***KUBECONFIG*** ကိုလဲ config file အစား အသုံး ပြုနိုင်သည်။

Let's SETUP

Kops ကို ဒီ link တွင် download ပြုလုပ်နိုင်ပါသည်။

Terminal

```
$ wget https://github.com/kubernetes/kops/releases/download/1.14.0-alpha.2/kops-linux-amd64

$ chmod +x kops-linux-amd64

$ sudo mv kops-linux-amd64 /usr/local/bin/kops
```

Kops ကို အသုံးပြုရန် AWS ၏ IAM user တစ်ယောက် လိုအပ်ပါသည်။ ထို IAM user တွင် Kops မှ AWS ၏ resource များကို အသုံးပြုရန် အနည်းဆုံး အောက်ပါ permission များ လိုအပ်ပါသည်။

- AmazonEC2FullAccess
- AmazonRoute53FullAccess
- AmazonS3FullAccess
- IAMFullAccess
- AmazonVPCFullAccess

လိုအပ်သော Permission နှင့် User တစ်ယောက်ဆောက်ပြီးသည့်အခါတွင် အောက်ဖော်ပြပါ Command နဲ့ AWS CLI ကို Setup လုပ်ပေးရမည်။

Terminal

```
$ aws configure
```

AWS CLI ကို setup လုပ်ပြီသောအခါ AWS ၏ S3 bucket တစ်ကို kops state file များ သိမ်းရန် အောက်ပါ command များကို အသုံးပြု၍ တည်ဆောက်နိုင်ပါသည်။

Terminal

```
$ export REGION=<your desire region>

$ export NAME=<your desire cluster name>

$ export KOPS_STATE_STORE=s3://${NAME} #your s3 bucket name

$ aws s3 mb $KOPS_STATE_STORE # create s3 bucket with your cluster name

$ aws s3api put-bucket-versioning --bucket ${NAME}  --versioning-configuration
Status=Enabled

$ aws s3api put-bucket-encryption --bucket ${NAME} --server-side-encryption-configuration
'{"Rules":[{"ApplyServerSideEncryptionByDefault":{"SSEAlgorithm":"AES256"}}]}'
```

AWS S3 တည်ဆောက်ပြီသောအခါ မာကျွန်တော်တို့ လိုချင်သည့် ပုံစံဖြင့် kubernetes cluster ကို တည်ဆောက်ရန် အဆင်သင့် ဖြစ်နေပြီပဲ ဖြစ်ပါသည်။ Kops ကို အသုံးပြု၍ private နဲ့ public topology ၂ မျိုးဖြင့် cluster ကို တည်ဆောက် နိုင်ပါသည်။

ဒီ scenario မှာတော့ private topology ကျွန်တော် အသုံးပြုသွားမှာ ဖြစ်သည်။ t3.medium instance type နှင့် weave network plugin ကို အသုံးပြုပြီး masters ၃ လုံး၊ nodes ၃လုံးကို အသုံးပြု၍တည်ဆောက်သွားမည်ဖြစ်ပါသည်။

Kops supports the following topologies on AWS

Topology	Value	Description
Public Cluster	public	All masters/nodes will be launched in a **public subnet** in the VPC
Private Cluster	private	All masters/nodes will be launched in a **private subnet** in the VPC

ထို.နောက် အောက်ဖော်ပြပါ Command နဲ့ လိုအပ်သော setup များဖြင့် cluster ကိုတည်ဆောက်ပြီး၊ထို cluster ရဲ့ config များကို ပြန်အသုံးပြုနိုင်ရန် **mycluster.yml** ဖိုင် ထဲကို standard output ဖြင့်ထည့်လိုက်ပါမယ်။

Terminal

```
$ kops create cluster ${NAME} --state=${KOPS_STATE_STORE} --master-count 3
--master-size t3.medium --node-count 3 --node-size t3.medium --zones
```

us-east-1a,us-east-1b,us-east-1c --topology private --networking weave --ssh-public-key ~/.ssh/id_rsa.pub -o yaml >> mycluster.yaml

ထို.နောက် **mycluster.yml** ဖိုင်ကို edit လုပ်ပီး additional policies section တွင် AWS IAM Policy မှ ELB and Route 53 policy များကိုပြင်ဆင်ပြီးနောက် Cluster Authentication အတွက် Line number ၁၀ နှင့် ၁၁ တွင်ရှိသော အောက်ဖော်ပြပါ **authentication:aws:{}** ကိုထည့်ပေးရန်လိုအပ်ပါသည်။

AWS IAM နှင့် Authentication ကို Integrate မလုပ်ချင်ပါက ယခု Config ကို ပြင်ဆင်စရာမလိုအပ်ပါ။ပြီးတော့ aws-iam-authenticator ကို ထည့်သွင်းပေးရန်မလိုပါ။

mycluster.yaml

```
apiVersion: kops/v1alpha2

kind: Cluster

metadata:

 creationTimestamp: null

 name: <your cluster name> # no need to touch

spec:

 api:

   loadBalancer:

     type: Public

 authentication:      #need to add this [authenication]

   aws: {}            #need to add this [aws: {}]

 authorization:

   rbac: {}

 channel: stable

 cloudProvider: aws

 configBase: s3://<your s3 name>/<your cluster name> #no need to touch

 etcdClusters:
```

```
- cpuRequest: 200m
  etcdMembers:
  - instanceGroup: master-ap-southeast-1a
    name: a
  - instanceGroup: master-ap-southeast-1b
    name: b
  - instanceGroup: master-ap-southeast-1c
    name: c
  memoryRequest: 100Mi
  name: main
- cpuRequest: 100m
  etcdMembers:
  - instanceGroup: master-ap-southeast-1a
    name: a
  - instanceGroup: master-ap-southeast-1b
    name: b
  - instanceGroup: master-ap-southeast-1c
    name: c
  memoryRequest: 100Mi
  name: events
iam:
  allowContainerRegistry: true
  legacy: false
kubelet:
```

```
  anonymousAuth: false
kubernetesApiAccess:
- 0.0.0.0/0
kubernetesVersion: 1.14.1
masterPublicName: api.yourcluster.com  #no need to touch
networkCIDR: 172.20.0.0/16
networking:
  weave:
    mtu: 8912
nonMasqueradeCIDR: 100.64.0.0/10
additionalPolicies:     #add this additional policy from this line to until line 160
  node: |
    [
      {
        "Effect": "Allow",
        "Action": [
          "acm:DescribeCertificate",
          "acm:ListCertificates",
          "acm:GetCertificate"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
```

```
  "Action": [
    "ec2:AuthorizeSecurityGroupIngress",
    "ec2:CreateSecurityGroup",
    "ec2:CreateTags",
    "ec2:DeleteTags",
    "ec2:DeleteSecurityGroup",
    "ec2:DescribeInstances",
    "ec2:DescribeInstanceStatus",
    "ec2:DescribeSecurityGroups",
    "ec2:DescribeSubnets",
    "ec2:DescribeTags",
    "ec2:DescribeVpcs",
    "ec2:ModifyInstanceAttribute",
    "ec2:ModifyNetworkInterfaceAttribute",
    "ec2:RevokeSecurityGroupIngress"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "elasticloadbalancing:AddTags",
    "elasticloadbalancing:CreateListener",
    "elasticloadbalancing:CreateLoadBalancer",
```

"elasticloadbalancing:CreateRule",

"elasticloadbalancing:CreateTargetGroup",

"elasticloadbalancing:DeleteListener",

"elasticloadbalancing:DeleteLoadBalancer",

"elasticloadbalancing:DeleteRule",

"elasticloadbalancing:DeleteTargetGroup",

"elasticloadbalancing:DeregisterTargets",

"elasticloadbalancing:DescribeListeners",

"elasticloadbalancing:DescribeLoadBalancers",

"elasticloadbalancing:DescribeLoadBalancerAttributes",

"elasticloadbalancing:DescribeRules",

"elasticloadbalancing:DescribeSSLPolicies",

"elasticloadbalancing:DescribeTags",

"elasticloadbalancing:DescribeTargetGroups",

"elasticloadbalancing:DescribeTargetGroupAttributes",

"elasticloadbalancing:DescribeTargetHealth",

"elasticloadbalancing:ModifyListener",

"elasticloadbalancing:ModifyLoadBalancerAttributes",

"elasticloadbalancing:ModifyRule",

"elasticloadbalancing:ModifyTargetGroup",

"elasticloadbalancing:ModifyTargetGroupAttributes",

"elasticloadbalancing:RegisterTargets",

"elasticloadbalancing:RemoveTags",

"elasticloadbalancing:SetIpAddressType",

```
    "elasticloadbalancing:SetSecurityGroups",
    "elasticloadbalancing:SetSubnets",
    "elasticloadbalancing:SetWebACL"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "iam:GetServerCertificate",
    "iam:ListServerCertificates"
  ],
  "Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "waf-regional:GetWebACLForResource",
    "waf-regional:GetWebACL",
    "waf-regional:AssociateWebACL",
    "waf-regional:DisassociateWebACL"
  ],
  "Resource": "*"
},
```

```
      {
        "Effect": "Allow",
        "Action": [
          "tag:GetResources",
          "tag:TagResources"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "waf:GetWebACL"
        ],
        "Resource": "*"
      },
      {
        "Effect": "Allow",
        "Action": [
          "route53:*"
        ],
        "Resource": "*"
      }
    ]
sshAccess:
```

- 0.0.0.0/0
subnets:
- cidr: 172.20.32.0/19
  name: ap-southeast-1a
  type: Private
  zone: ap-southeast-1a
- cidr: 172.20.64.0/19
  name: ap-southeast-1b
  type: Private
  zone: ap-southeast-1b
- cidr: 172.20.96.0/19
  name: ap-southeast-1c
  type: Private
  zone: ap-southeast-1c
- cidr: 172.20.0.0/22
  name: utility-ap-southeast-1a
  type: Utility
  zone: ap-southeast-1a
- cidr: 172.20.4.0/22
  name: utility-ap-southeast-1b
  type: Utility
  zone: ap-southeast-1b
- cidr: 172.20.8.0/22
  name: utility-ap-southeast-1c

```
    type: Utility
    zone: ap-southeast-1c
  topology:
    dns:
      type: Public
    masters: private
    nodes: private

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
  creationTimestamp: null
  labels:
    kops.k8s.io/cluster: hearu.org
  name: master-ap-southeast-1a
spec:
  image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
  machineType: t3.medium
  maxSize: 1
  minSize: 1
  nodeLabels:
    kops.k8s.io/instancegroup: master-ap-southeast-1a
```

```
 role: Master
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1a

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com # no need to touch
 name: master-ap-southeast-1b
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 1
 minSize: 1
 nodeLabels:
   kops.k8s.io/instancegroup: master-ap-southeast-1b
 role: Master
 rootVolumeSize: 60
```

```
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1b

---

apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com #no need to touch
 name: master-ap-southeast-1c
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 1
 minSize: 1
 nodeLabels:
   kops.k8s.io/instancegroup: master-ap-southeast-1c
 role: Master
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
```

```
 - ap-southeast-1c


---


apiVersion: kops/v1alpha2
kind: InstanceGroup
metadata:
 creationTimestamp: null
 labels:
   kops.k8s.io/cluster: yourcluster.com #no need to touch
 name: nodes
spec:
 image: kope.io/k8s-1.12-debian-stretch-amd64-hvm-ebs-2019-05-13
 machineType: t3.medium
 maxSize: 3
 minSize: 3
 nodeLabels:
   kops.k8s.io/instancegroup: nodes
 role: Node
 rootVolumeSize: 60
 rootVolumeType: gp2
 subnets:
 - ap-southeast-1a
 - ap-southeast-1b
```

- ap-southeast-1c

အားလုံးပြင်ဆင်ပြီးနောက် cluster ကို အောက်ပါ command များဖြင့် bootstrap လုပ်နိုင်ပါပြီ။ Bootstrap လုပ်နေချိန်တွင် မိနစ်အနည်းငယ် စောင့်ရန်လိုအပ်ပါသည်။

Terminal

```
$ kops create -f mycluster.yaml

$ kops create secret --name ${NAME} sshpublickey admin -i ~/.ssh/id_rsa.pub

$ kops update cluster ${NAME} --yes

$ kops kops validate cluster # validate cluster is check for your cluster state
```

Cluster တည်ဆောက်ပြီးဖြစ်သော်လည်း Status ကတော့ valid ဖြစ်သေးမှာမဟုတ်ပါဘူး။ ဘာလို့လဲဆိုတော့ authentication plugin ၏ configuration လိုနေသောကြောင့်ဖြစ်ပါသည်။

json

```
//K8sAdmin

{

   "Version": "2012-10-17",

   "Statement": [

       {

           "Sid": "VisualEditor0",

           "Effect": "Allow",

           "Action": [

               "sts:DecodeAuthorizationMessage",

               "sts:GetCallerIdentity"

           ],

           "Resource": "*"
```

```
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:*",
            "Resource": "arn:aws:iam::<account id>:role/K8sAdmin"
        }
    ]
}
```

json

//K8sDeveloper

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::<account id:role/K8sDeveloper"
        },
        {
            "Sid": "VisualEditor1",
            "Effect": "Allow",
            "Action": "sts:GetCallerIdentity",
```

```
        "Resource": "*"
      }
  ]
}
```

အောက်တွင်ဖော်ပြထားသော commands ဖြင့် K8sDeveloper နှင့် K8sAdmin အစရှိသော role နှစ်ခုကို configmap အတွက် တည်ဆောက်သည့်အခါတွင် အပေါ်မှာ json များထွက်လာမည်ဖြစ်ပါသည်။

Example ပြထားခြင်းဖြစ်ပါသည်။

Terminal

```
$ export ACCOUNT_ID=$(aws sts get-caller-identity --output text --query 'Account')

$ export DEVPOLICY=$(echo -n
'{"Version":"2012-10-17","Statement":[{"Sid":"VisualEditor0","Effect":"Allow","Action":["sts:Decod
eAuthorizationMessage","sts:GetCallerIdentity"],"Resource":"*"},{"Sid":"VisualEditor1","Effect":"A
llow","Action":"sts:*","Resource":"arn:aws:iam::'; echo -n "$ACCOUNT_ID"; echo -n
':role/K8sDeveloper"}]}')

$ aws iam create-role \

 --role-name K8sDeveloper \

 --description "Kubernetes developer role (for AWS IAM Authenticator for AWS)." \

 --assume-role-policy-document "$DEVPOLICY" \

 --output text \

 --query 'Role.Arn'

$ export ADMPOLICY=$(echo -n
'{"Version":"2012-10-17","Statement":[{"Sid":"VisualEditor0","Effect":"Allow","Action":["sts:Decod
eAuthorizationMessage","sts:GetCallerIdentity"],"Resource":"*"},{"Sid":"VisualEditor1","Effect":"A
llow","Action":"sts:*","Resource":"arn:aws:iam::'; echo -n "$ACCOUNT_ID"; echo -n
':role/K8sAdmin"}]}')

$ aws iam create-role \
```

```
 --role-name K8sAdmin \

 --description "Kubernetes administrator role (for AWS IAM Authenticator for AWS)." \

 --assume-role-policy-document "$ADMPOLICY" \

 --output text \

 --query 'Role.Arn'
```

ကျွန်တော်တို့. IAM role နှင့် policies များကိုတည်ဆောက်ပြီးနောက် Kubernets resources များတွက် authorization ကို configure ချပေးရန်လိုအပ်ပါသည်။

ထို.နောက် RBAC authorization ကို အသုံးပြုပြီး developer role အတွက် production နှင့် development namespace နှစ်ခုကို configure ချပေးထားပါသည်။

K8sDeveloper role အတွင်းရှိသော user များသည် ထို namespace နှစ်ခုနှင့် resource အချို့.သာလျင် အသုံးပြုနိုင်မည်ဖြစ်သည်။

developer-rbac.yaml

```
# RBAC Policy for developer

# create 2 Roles on for developer and binding roles

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: Role

metadata:

 namespace: development

 name: developer

rules:

- apiGroups: ["*"]

 resources:
["deployments","configmaps","pods","secrets","services","endpoints","persistentvolumeclaims","
events","namespaces","statefulsets","replicasets","jobs","cronjobs"]

 verbs: ["create","get","delete","list","update","edit","watch","exec","patch"]

- apiGroups:
```

```
 - '*'
 resources:
 - 'pods/exec'
 - 'pods/portforward'
 - 'services/portforward'
 verbs:
 - create
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
 name: developer
 namespace: development
roleRef:
 kind: Role
 name: developer
 apiGroup: rbac.authorization.k8s.io
subjects:
- kind: Group
 namespace: development
 name: k8s.developer
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: Role
```

```
metadata:

 namespace: production

 name: developer

rules:

- apiGroups: ["*"]

 resources:
["deployments","configmaps","pods","secrets","services","endpoints","persistentvolumeclaims","
events","namespaces","statefulsets","replicasets","jobs","cronjobs"]

 verbs: ["create","get","delete","list","update","edit","watch","exec","patch"]

- apiGroups:

 - '*'

 resources:

 - 'pods/exec'

 - 'pods/portforward'

 - 'services/portforward'

 verbs:

 - create

---

apiVersion: rbac.authorization.k8s.io/v1

kind: RoleBinding

metadata:

 name: developer

 namespace: production

roleRef:
```

```
 kind: Role

 name: developer

 apiGroup: rbac.authorization.k8s.io

subjects:

- kind: Group

 namespace: production

 name: k8s.developer
```

Terminal

```
$ kubectl create namespace development

$ kubectl create namespace production

$ kubectl create -f developer-rbac.yaml
```

aws-iam-authenticator အလုပ်လုပ်ရန် နင့် သတ်မှတ်ထားသည့် user group policy များ အသက်ဝင်ရန် အောက်ပါ configmap ကို create ပြုလုပ်ပေးရန်လိုအပ်ပါသည်။

aws-auth.yaml

```
apiVersion: v1

kind: ConfigMap

metadata:

 namespace: kube-system

 name: aws-iam-authenticator

 labels:

   k8s-app: aws-iam-authenticator

data:

 config.yaml: |

   clusterID: api.mycluster.com
```

```
  server:
    mapRoles:
    - roleARN: arn:aws:iam::<account id>:role/K8sAdmin
      username: kubernetes-admin
      groups:
      - system:masters
    - roleARN: arn:aws:iam::<account id>:role/K8sDeveloper
      username: kubernetes-developer
      groups:
      - k8s.developer
```

Terminal

```
$ kubectl create -f aws-auth.yaml
```

ယခုအခြေအနေတွင် Kubernetes cluster နှင့် authentication/authorization setup ပြုလုပ်ခြင်းမှာ ပြီးမြောက်လုနီးပါးဖြစ်ပါသည်။ Kubernetes client (kubectl) မှ cluster ကို access လုပ်ရန် configure ချဖို. လိုအပ်ပါသည်။ အောက်ဖော်ပြပါ config နှစ်ခုသည် Developer နှင့် Administrator အတွက် ဖြစ်ပါသည်။ ကျွန်တော် ကတော့ Admin config အသုံးပြုပြီး AWS Application LoadBalancer ကိုဆက်လက် setup လုပ်သွားမှာ ဖြစ်ပါသည်။

~/kube/.config (Admin)

```
# this is for Kubernetes Administrator

# you must change this the value of 'certificate-authority-data' with your ca.

apiVersion: v1

clusters:

- cluster:

  certificate-authority-data:
LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMwekNDQWJ1Z0F3SUJBZ0lNRmFlcH
Bua0lEUnRwNkpub01BMEdDU3FHU0liM0RRRUJDd1VBTUJVeEV6QVIKQmdOVkJBTVRDbX
QxWW1WeWJtVjBaWE13SGhjTk1Ua3dOakV4TURVd05USTRXaGNOTWprd05qRXdNRFV3Tl
RJNApXakFWTVJNd0VRWURWUVFERXdwcmRXSmxjbTVsZEdWek1JSUJJakFOQmdrcWhra
```

Uc5dzBCQVFFRkFBT0NBUThBCk1JSUJDZ0tDQVFFQXpmMzlOOUIvaGpUQnViM0tETEhzM1
NJUmhVL2pVcUt2RWhTWnpCQngwR0tXNmtONmRUVkkKK3dwUnRoeU9Yd2ZWRHA4RmxR
YUFYbWt0UGtGTG9jU2VSelBUWXRod1VBamhSbVNoZFJtcW1yaXBZTlhFdEZYMAp6SkZFS
2NRNDJ0TEhYanZZbVc3SGxYc0FDSDRCKy82SEtwVmxGSVdNTDAwb2tTUmEzZVBvdTdW
YzlqTkZhd3R0CmFoMThrNStLcHVoL0FyUEs2dmdhcEhLclB2dUNaZG9FWkNtaE4rSVdZTjN6
Uk1Wb1A1TEVoTEx5RFFzUzYvN0sKQ3prQ09Pam1QWW01UmxnOVRvb25ZcStwbS9LMXc4
V1NLVnZick9VZXF0dXkxUlJYWUxtc0VNVmx0NDRQSS9YTgpEeUlubmtpVmRwc1A4ejAwRjZ
5MGEwYXFpT3RGUVVyd3NRSURBUUFCb3lNd0lUQU9CZ05WSFE4QkFmOEVCQU1DCkFR
WXdEd1lEVlIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFq
Um5ndnJ6SmRMWm8KRFdlbVFaMi83N0xLTmkzTmpRQU9NZUg3WXVpZnlNbTZNdTB2V3F6
UU54SDJ1TGhFd0JRN0VocEpYSnFoMmp0dQpldnR6YUF6N2o5MEVKdlN6WHRtSDRQaHd1
NFZFMHJWRGVlbUhrMXRZRjhsQ3hVM2xtZFRGZjJ4M29qVnVKNndpCkd3c3RaU0s5V2J5V
W53RjNLUXdkMndmd3VUVVpSRGtuSkI2ZGhOcUxNakN3V1M1ekpDUjcvYXFuM0tqb0xvWEg
KWkI1VW1UcGd2Sm9rWm0wcUF0ODRNS0hFb2ZCYXJ0M1BUcnplWGI0N1VHRlMrM0h1anV
QUTBreW1DT0crK1ZEMwo0K0M5eVlXaTRhYnd6SE9rTmQvMlFKZVoyQjhjbjhlVU5kUGFSSnk
weS9keVpSd0NZdXFKSkxZeTVGUTFiQlUwCm96bThnMG01ZWc9PQotLS0tLUVORCBDRVJ
USUZJQ0FURS0tLS0tCg==

  server: https://api.yourcluster.com

 name: yourcluster.com

contexts:

- context:

  cluster: yourcluster.com

  user: kubernetes-admin

 name: aws

current-context: aws

kind: Config

preferences: {}

users:

- name: kubernetes-admin

 user:

  exec:

```
    apiVersion: client.authentication.k8s.io/v1alpha1

    args:

    - token

    - -i

    - api.yourcluster.com

    - -r

    - arn:aws:iam::<account id>:role/K8sAdmin

    command: aws-iam-authenticator

    env: null
```

~/kube/.config (Developer)

```
# this is for Kubernetes Developer

# you must change this the value of 'certificate-authority-data' with your ca.

apiVersion: v1

clusters:

- cluster:

  certificate-authority-data:
```


LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JSUMwekNDQWJ1Z0F3SUJBZ0lNRmFlcH
Bua0lEUnRwNkpub01BMEdDU3FHU0liM0RRRUJDd1VBTUJVeEV6QVIKQmdOVkJBTVRDbX
QxWW1WeWJtVjBaWE13SGhjTk1Ua3dOakV4TURVd05USTRXaGNOTWprd05qRXdNRFV3Tl
RJNApXakFWTVJNd0VRWURWUVFERXdwcmRXSmxjbTVsZEdWek1JSUJJakFOQmdrcWhra
Uc5dzBCQVFFRkFBT0NBUThBCk1JSUJDZ0tDQVFFQXpmMzlOOUIvaGpUQnViM0tETEhzM1
NJUmhVL2pVcUt2RWhTWnpCQngwR0tXNmtONmRUVkkKK3dwUnRoeU9Yd2ZWRHA4RmxR
YUFYbWt0UGtGTG9jU2VSelBUWXRod1VBamhSbVNoZFJtcW1yaXBZTlhFdEZYMAp6SkZFS
2NRNDJ0TEhYanZZbVc3SGxYc0FDSDRCKy82SEtwVmxGSVdNTDAwb2tTUmEzZVBvdTdW
YzlqTkZhd3R0CmFoMThrNStLcHVoL0FyUEs2dmdhcEhLclB2dUNaZG9FWkNtaE4rSVdZTjN6
Uk1Wb1A1TEVoTEx5RFFzUzYvN0sKQ3prQ09Pam1QWW01UmxnOVRvb25ZcStwbS9LMXc4
V1NLVnZick9VZXF0dXkxUlJYWUxtc0VNVmx0NDRQSS9YTgpEeUlubmtpVmRwc1A4ejAwRjZ
5MGEwYXFpT3RGUVVyd3NRSURBUUFCb3lNd0lUQU9CZ05WSFE4QkFmOEVCQU1DCkFR
WXdEd1lEVIIwVEFRSC9CQVV3QXdFQi96QU5CZ2txaGtpRzl3MEJBUXNGQUFPQ0FRRUFq
Um5ndnJ6SmRMWm8KRFdlbVFaMi83N0xLTmkzTmpRQU9NZUg3WXVpZnlNbTZNdTB2V3F6
UU54SDJ1TGhFd0JRN0VocEpYSnFoMmp0dQpldnR6YUF6N2o5MEVKdlN6WHRtSDRQaHd1

NFZFMHJWRGVlbUhrMXRZRjhsQ3hVM2xtZFRGZjJ4M29qVnVKNndpCkd3c3RaU0s5V2J5V
W53RjNLUXdkMndmd3VUVVpSRGtuSkI2ZGhOcUxNakN3V1M1ekpDUjcvYXFuM0tqb0xvWEg
KWkI1VW1UcGd2Sm9rWm0wcUF0ODRNS0hFb2ZCYXJ0M1BUcnplWGI0N1VHRlMrM0h1anV
QUTBreW1DT0crK1ZEMwo0K0M5eVlXaTRhYnd6SE9rTmQvMlFKZVoyQjhjbjhlVU5kUGFSSnk
weS9keVpSd0NZdXFKSkxZeTVGUTFiQlUwCm96bThnMG01ZWc9PQotLS0tLUVORCBDRVJ
USUZJQ0FURS0tLS0tCg==

```
   server: https://api.yourcluster.com

 name: yourcluster.com

contexts:

- context:

   cluster: yourcluster.com

   namespace: development  #you will notiace this. This is means developere default
namesapce is development.

   user: kubernetes-developer

 name: aws

current-context: aws

kind: Config

preferences: {}

users:

- name: kubernetes-developer

 user:

   exec:

     apiVersion: client.authentication.k8s.io/v1alpha1

     args:

     - token

     - -i
```

```
    - api.yourcluster.com

    - -r

    - arn:aws:iam::<account id>:role/K8sDeveloper

    command: aws-iam-authenticator

    env: null
```

Terminal

```
$ wget
https://github.com/kubernetes-sigs/aws-iam-authenticator/releases/download/v0.4.0/aws-iam-authenticator_0.4.0_linux_amd64

$ chmod +x aws-iam-authenticator_0.4.0_linux_amd64

$ sudo mv aws-iam-authenticator_0.4.0_linux_amd64 /usr/local/bin/aws-iam-authenticator

$ aws-iam-authenicator version

{"Version":"0.4.0","Commit":"9999cd8047782c94c08e2d24b77dc9488947aa03"}

$ kubectl get nodes # test what you authorized
```

ထို့နောက် AWS Application LoadBalancer ကို Kubernetes Ingress Controller နဲ့ integrate ပြုလုပ်ရန် အောက်ဖော်ပြပါအတိုင်း RBAC နှင့် alb-ingress-controller deployment ကို kube-system တွင် create လုပ်ရမည်ဖြစ်ပါသည်။ လိုအပ်သော additional policy များကို cluster bootstrap လုပ်ခဲ့တုန်း ကထည့်သွင်းခဲ့ပြီးဖြစ်ပါသည်။

alb-rbac.yaml

```
apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRole

metadata:

 labels:

   app: alb-ingress-controller

 name: alb-ingress-controller

rules:
```

```
- apiGroups:
    - ""
    - extensions
  resources:
    - configmaps
    - endpoints
    - events
    - ingresses
    - ingresses/status
    - services
  verbs:
    - create
    - get
    - list
    - update
    - watch
    - patch
- apiGroups:
    - ""
    - extensions
  resources:
    - nodes
    - pods
    - secrets
```

```
    - services
    - namespaces
  verbs:
    - get
    - list
    - watch
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress-controller
roleRef:
 apiGroup: rbac.authorization.k8s.io
 kind: ClusterRole
 name: alb-ingress-controller
subjects:
 - kind: ServiceAccount
   name: alb-ingress
   namespace: kube-system
---
apiVersion: v1
kind: ServiceAccount
```

```
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress
 namespace: kube-system
```

alb-ingress-controller.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
 labels:
   app: alb-ingress-controller
 name: alb-ingress-controller
 namespace: kube-system
spec:
 replicas: 1
 selector:
   matchLabels:
     app: alb-ingress-controller
 strategy:
   rollingUpdate:
     maxSurge: 1
     maxUnavailable: 1
   type: RollingUpdate
 template:
```

```
  metadata:
    creationTimestamp: null
    labels:
      app: alb-ingress-controller
  spec:
    containers:
      - args:
          - --ingress-class=alb
          - --cluster-name=yourcluster.com
        env:
        image:
894847497797.dkr.ecr.us-west-2.amazonaws.com/aws-alb-ingress-controller:v1.0.0
        imagePullPolicy: Always
        name: server
        resources: {}
        terminationMessagePath: /dev/termination-log
    dnsPolicy: ClusterFirst
    restartPolicy: Always
    securityContext: {}
    terminationGracePeriodSeconds: 30
    serviceAccountName: alb-ingress
    serviceAccount: alb-ingress
```

Terminal

```
# setup with your administrator account.
```

```
$ kubectl create -f alb-rbac.yaml

$ kubectl create -f alb-ingress-controller.yaml

$ kubectl get pod -n kube-system
```

Ingress controller setup လုပ်ပီးနောက် external DNS အတွက် DNS records များကို Ingress create လုပ်လိုက်သည်နှင့် တပြိုင်နက်ထဲ AWS route 53 တွင် automatically update ပြုလုပ်သွားရန် အောက်ဖော်ပြပါ service account, rbac for external dns, deployment များကို ဆက်လက်တည်ဆောက်ရမည်ဖြစ်သည်။

external-dns.yaml

```
apiVersion: v1

kind: ServiceAccount

metadata:

 name: external-dns

---

apiVersion: rbac.authorization.k8s.io/v1beta1

kind: ClusterRole

metadata:

 name: external-dns

rules:

- apiGroups: [""]

 resources: ["services"]

 verbs: ["get","watch","list"]

- apiGroups: [""]

 resources: ["pods"]

 verbs: ["get","watch","list"]

- apiGroups: ["extensions"]
```

```
  resources: ["ingresses"]
  verbs: ["get","watch","list"]
- apiGroups: [""]
  resources: ["nodes"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1beta1
kind: ClusterRoleBinding
metadata:
  name: external-dns-viewer
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: external-dns
subjects:
- kind: ServiceAccount
  name: external-dns
  namespace: kube-system
---
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: external-dns
spec:
```

```
 strategy:
   type: Recreate
 template:
   metadata:
     labels:
       app: external-dns
   spec:
     serviceAccountName: external-dns
     containers:
     - name: external-dns
       image: registry.opensource.zalan.do/teapot/external-dns:v0.5.9
       args:
       - --source=service
       - --source=ingress
       - --domain-filter=yourcluster.com # will make ExternalDNS see only the hosted zones matching provided domain, omit to process all available hosted zones
       - --provider=aws
       - --policy=upsert-only # would prevent ExternalDNS from deleting any records, omit to enable full synchronization
       - --aws-zone-type=public # only look at public hosted zones (valid values are public, private or no value for both)
       - --registry=txt
       - --txt-owner-id=my-identifier
```

Terminal

$ kubectl create -f external-dns.yaml

ယခုအခြေအနေတွင်အားလုံးတည်ဆောက်ပြီး ဖြစ်ပါသည်။

တည်ဆောက်ခဲ့သည့် Cluster, IAM authentication and k8s RBAC, ALB ingress and automatic external DNS service များကို စမ်းသပ်ရန်အတွက် အောက်ဖော်ပြပါ 2048 web application game ဖြင့်စမ်းသပ်နိုင်ပါသည်။

Terminal

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-namespace.yaml

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-deployment.yaml

$ kubectl apply -f https://raw.githubusercontent.com/kubernetes-sigs/aws-alb-ingress-controller/v1.0.0/docs/examples/2048/2048-service.yaml

$ kubectl get po,deploy,svc -n 2048-game

2048-ingress.yaml

```
apiVersion: extensions/v1beta1

kind: Ingress

metadata:

 name: "2048-ingress"

 namespace: "2048-game"

 annotations:

   kubernetes.io/ingress.class: alb

   alb.ingress.kubernetes.io/scheme: internet-facing

 labels:

   app: 2048-ingress
```

```
spec:
 rules:
 - host: yourcluster.com
   http:
     paths:
     - path: /*
       backend:
         serviceName: "service-2048"
         servicePort: 80
```

Terminal

```
$ kubectl create -f 2048-ingress.yaml
$ kubectl get ingress -n 2048-game
$ dig yourcluster.com # see and wait
$ google-chrome-stable yourcluster.com
```

ယခု ကျွန်တော်တို. လုပ်ပြထားသော tutorial လေးသည် Non-Profit Organization တစ်ခုအတွက် ကျွန်တော်တို. Setup လုပ်ပေးခဲ့သော Production Grade Kops Cluster Setup Use-case မှ ကောက်နုတ်ဖော်ပြထားခြင်းဖြစ်သည့်အတွက် စာဖတ်သူတို.စိတ်ထဲတွင် မရှင်းမလင်းတစ်ခုခုခံစားခဲ့ရရင်ကျွန်တော်တို.ရဲ့ Facebook Group or Page ကိုဆက်သွယ်မေးမြန်းနိုင်ပါသည်။

Amazon EKS Cluster Provisioning

Introduction to Amazon EKS - Kubernetes on AWS

ELASTIC CONTAINER SERVICE FOR
KUBERNETES (EKS)

Introduction to Amazon EKS - Kubernetes on AWS

ပထမဦးစွာ ဒီ Blog tutorial က AWS User Group Myanmar က ကျင်းပမဲ့ Online Meetup မှာ ကျွန်တော်ပြောသွားမဲ့ "**Amazon EKS - Kubernetes on AWS**" Topic လေးနဲ့တူတူ လေ့လာပြီး ကိုယ်ကိုတိုင် AWS EKS cluster ဆောက်ပြီးလိုက်စမ်းနိုင်အောင်ရည်ရွယ်ပြီးရေးထားချင်းဖြစ်ပါတယ်။

ကျွန်တော်တို့.ပွဲလေး ကိုတက်ရောက်ဖို့.ဖိတ်ပါရစေ .. ပြီးတော့ Topic များကိုလည်းအောက်မှာဖော်ပြထားတဲ့ Topic Board မှာ ဝင်ရောက်ကြည့်နိုင်ပါတယ်။

https://airtable.com/shrGToJczSuy1KZAr

အဲ နောက်တစ်ခုက Speaker အဖြစ်နဲ့ Sharing လုပ်ပေးချင်ရင်လည်း ကျွန်တော်တို့.က ဝမ်းမြောက်ဝမ်းသာ ဖိတ်ကြားပါရစေ။ အောက်က လင့် လေးကနေ စာရင်းသွင်းလိုရပါတယ်ဗျ။

https://airtable.com/shrV1vefDtNLcAaZ0

ပွဲအကြောင်းပြောပြီးပြီဆိုတော့ရေးမည့်ဟာလေးကိုဆက်ပါရစေ။

Amazon EKS အကြောင်းမပြောခင် ကနဦးအနေနဲ့ Kubernetes ဆိုတာဘာလဲ ဘယ်လို လဲ သူ.ရဲ့ architecture က ဘယ်လိုအလုပ်တွေလုပ်လဲနည်းနည်းအကျဉ်းချုံးကြည့်ရအောင်။ Kubernetes ရဲ့ official documentation မှာတော့အောက်ကအတိုင်းဖော်ပြထားပါတယ်။ အဓိက ကတော့ မြန်မာလိုပြောရရင်တော့ Kubernetes က containerized application တွေကို လွယ်လွယ်ကူကူ စီမံခံခွဲနိုင်အောင်၊ မောင်းနှင်းနိုင်အောင် ပြုလုပ်ထားတဲ့ solution တစ်ခုဖြစ်တယ်ပေါ့။

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

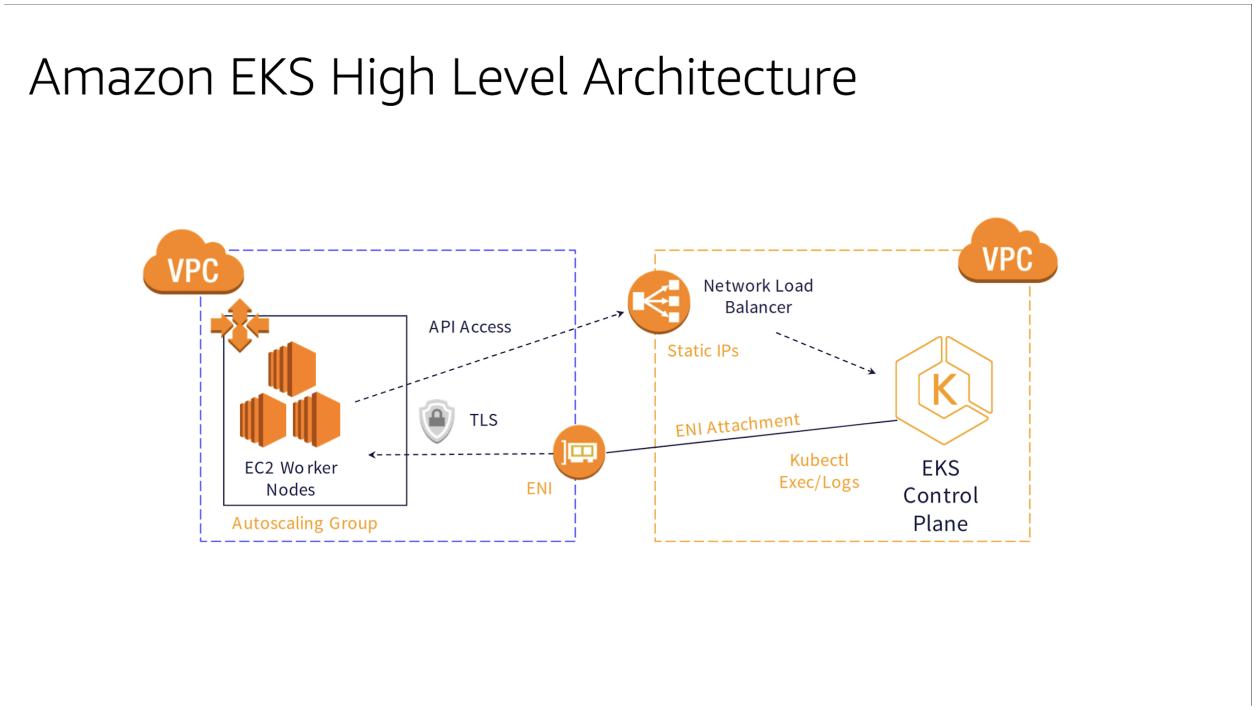
ဆိုတော့ Kubernetes Cluster ရဲ့ architecture, components တွေကိုလေ့လာကြည့်ရအောင်။Master (Control Plane) မှာဘာတွေပါလဲ၊ Worker node (Minion) မှာဘာတွေပါဝင်လဲ လေ့လာကြည့်ရအောင်။ ကျွန်တော်ဒီ Blog Post က Amazon EKS cluster provisioning ဆိုတော့ Kubernetes cluster architecture ကိုတော့ မရေးတော့ဘူး ဒါပေမဲ့ မြန်မာလိုပဲရေးထားပြီးနားလည်ရလွယ်ကူတဲ့ Blog content လေးကိုအောက်မှာ link အနေနဲ့ ညွှန်းလိုက်ပါတယ်။

https://blog.k8smm.org/k8s-cluster-achitecture

Amazon EKS က Amazon Web Services ရဲ. Managed Kubernetes Service တစ်ခုဖြစ်ပါသည်။ အဓိက အားဖြင့် Kubernetes ရဲ့ master control plane ကို AWS ဘက်က managed လုပ်ပေးချင်းကို ဆိုလိုတာပါ။ အဲ့ထက်ပိုပြီး EKS က data plane ဖြစ်တဲ့ worker nodes များပါ managed လုပ်ပေးနိုင်ပြီ ဟုလည်း AWS က announced လုပ်ထားပါတယ်။

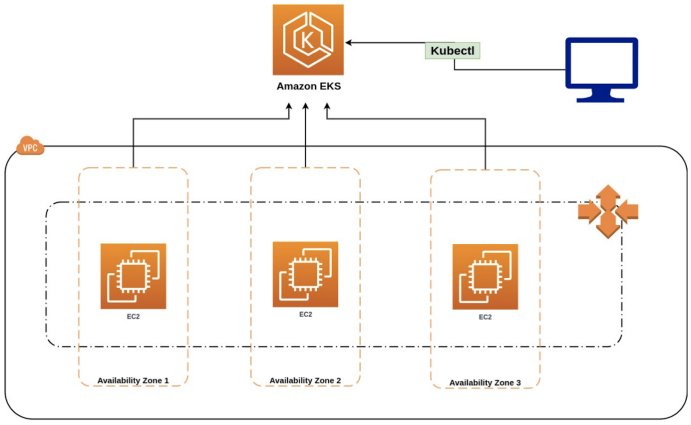
အိုကေ .. ဒါဆို လက်တွေ့ EKS Cluster ကို ဆောက်ကြည့်ကြရအောင်။အရင်ဆုံး EKS Cluster မဆောက်ခင် Official Documentation ကို Share ပါရစေ။

ပြီးနောက် AWS EKS ရဲ့ Architecture လေးကိုလည်း တစ်ချက်လောက်လေ့လာကြရအောင်။



EKS High Level Architecture

Amazon EKS (High Level Consumption Design Flow)



EKS Consumption Design (Re-Draw by Wai Yan Min)

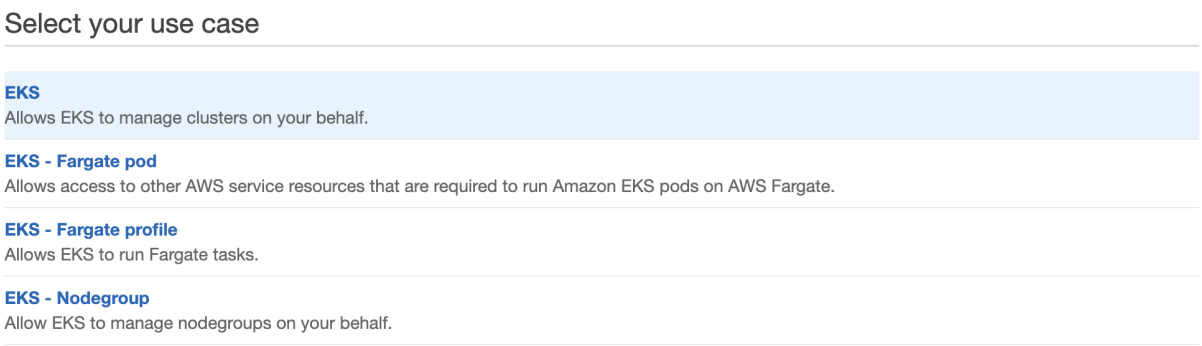
Official Documentation here 👀

https://docs.aws.amazon.com/eks/latest/userguide/what-is-eks.html

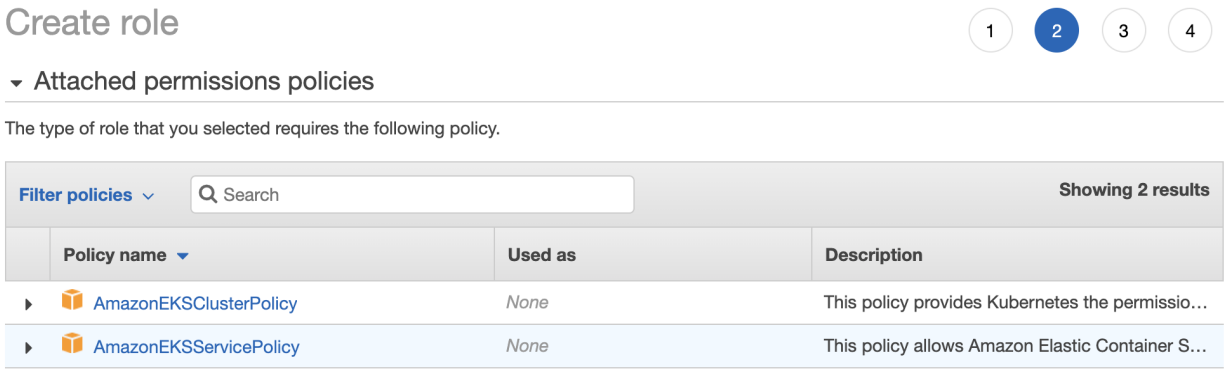
CloudFormation Template တွေကိုလည်းတနေရာထဲက Download ချင်ရင်လည်းအဆင်ပြေဖို.တွက်ပြန် Share ပေးထားပါတယ်။

https://github.com/phyominhtun1990/AWS-EKS-Cluster-Introduction

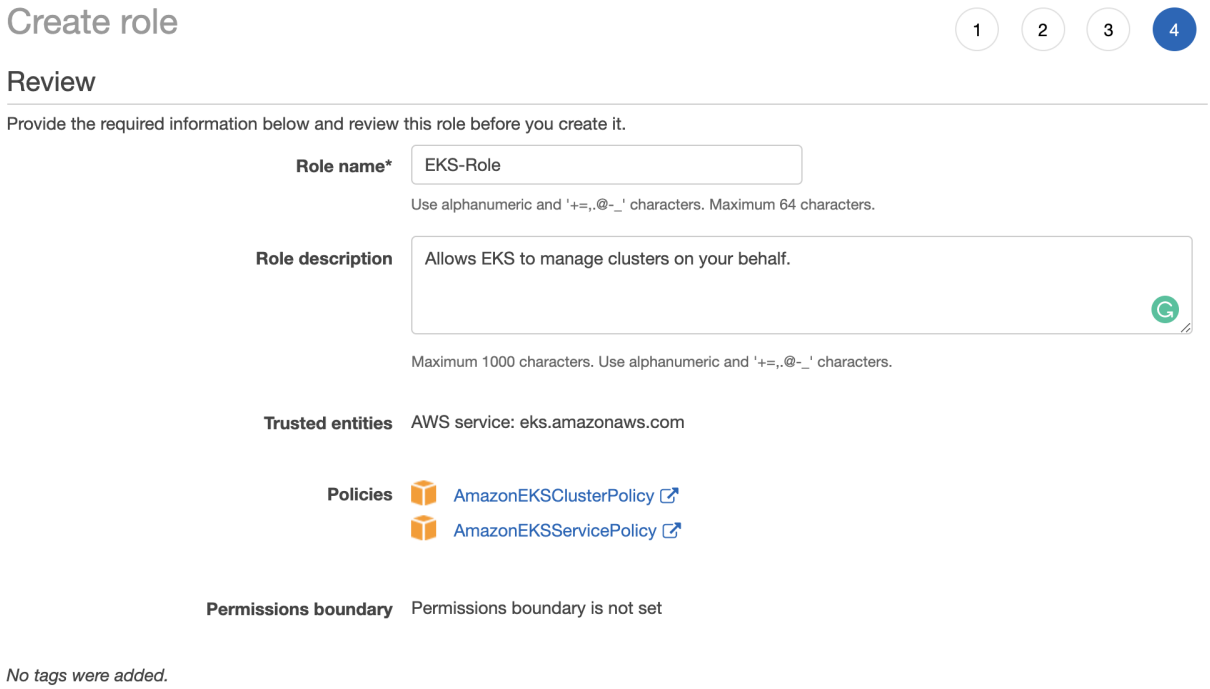
ဆိုတော့ကျွန်တော်တို. EKS Cluster စဆောက်ကြရအောင်။ပထမဆုံးအနေနဲ့ cluster provision မလုပ်ခင် EKS တွက် IAM Role ကို စဆောက်ရပါမယ်။ AWS IAM Console ကို သွားပြီးအောက်မှာဖော်ပြထားတဲ့အတိုင်း IAM Role တစ်ခုကို တည်ဆောက်ပါမယ်။



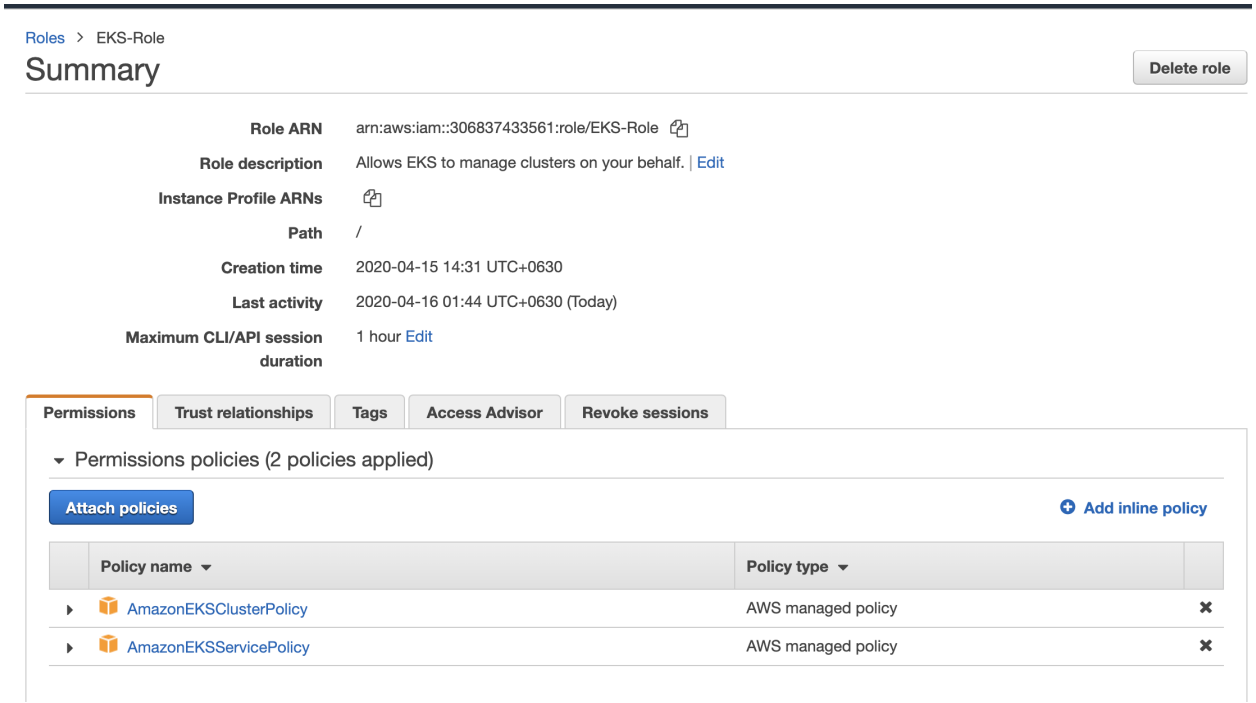
Creating EKS Role



Creating EKS Role



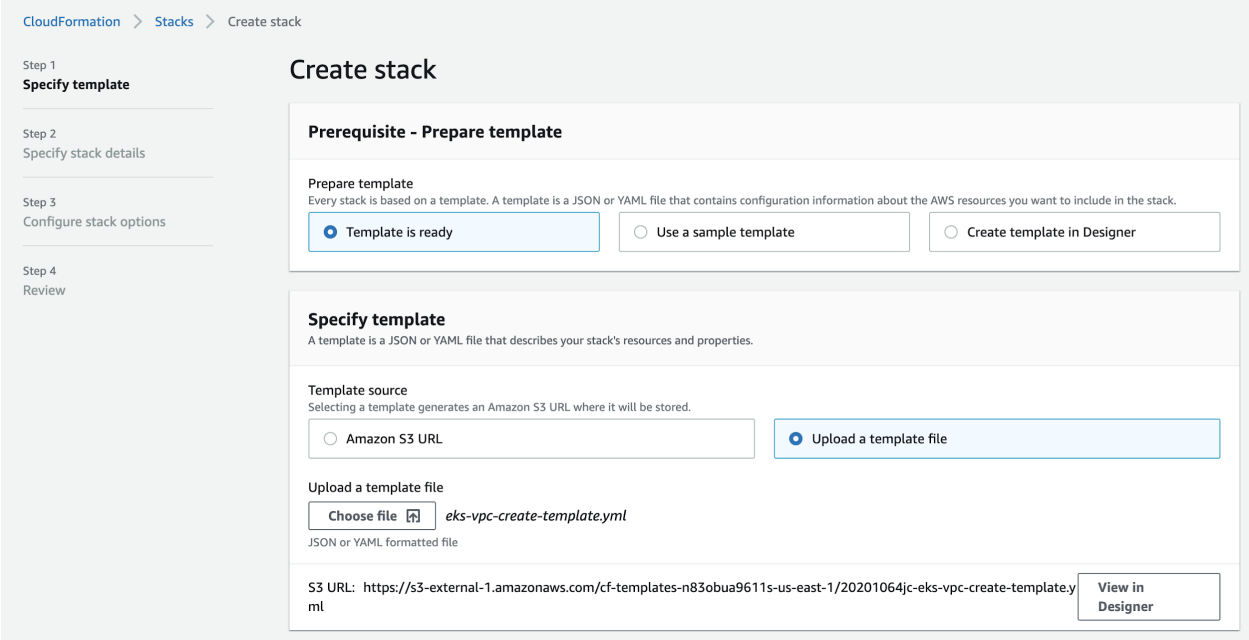
Creating EKS Role



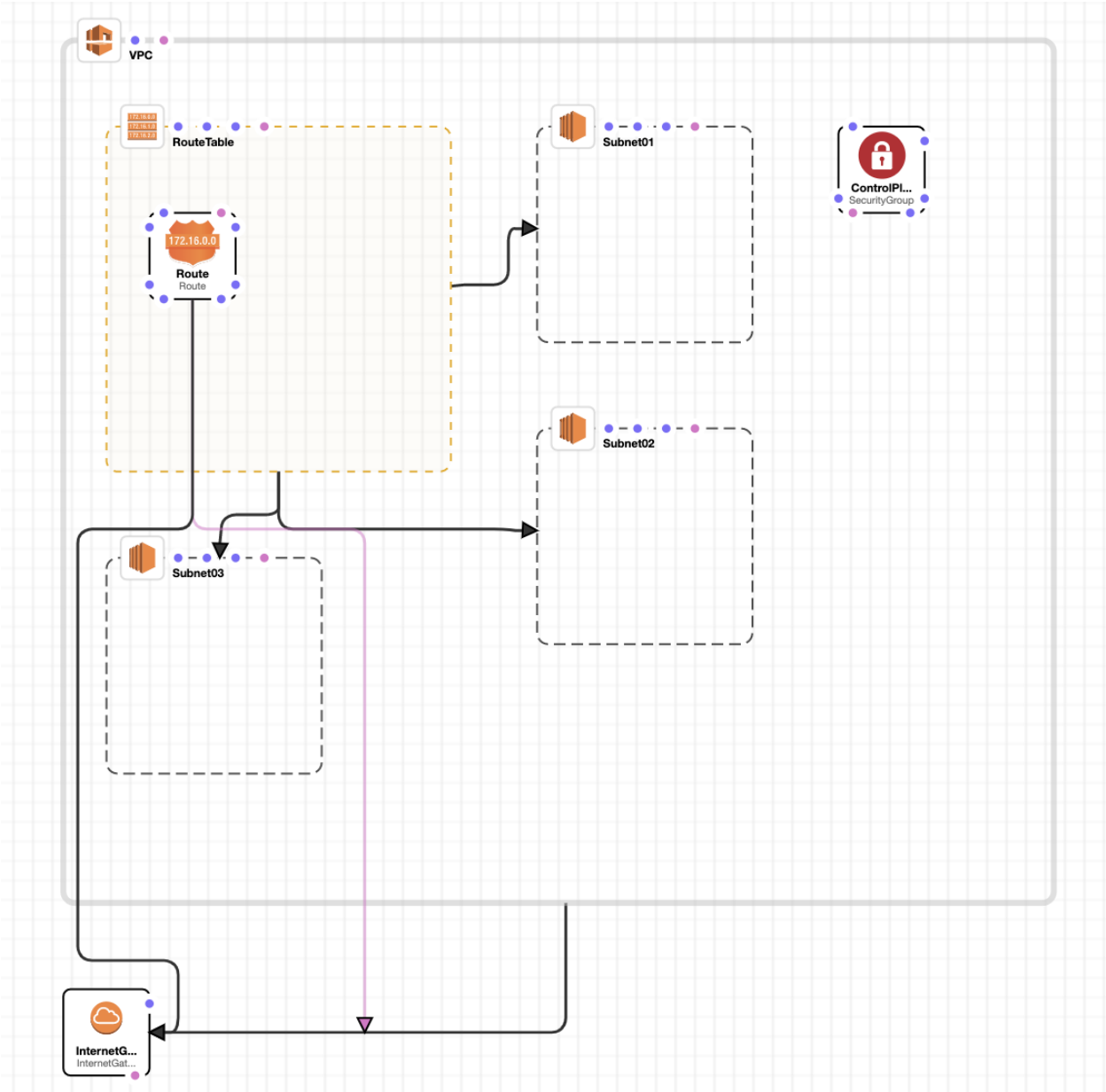
EKS IAM Role ဆောက်ပြီးပြီ ဆိုတော့ နောက်တစ်ဆင့်အနေနဲ့ EKS Cluster အတွက် VPC ဆောက်ပြီး Networking setting ချပေးရပါမယ်။ VPC ဆောက်ရာတွင် manual ဆောက်တာထက် CloudFormation နဲ့တည်ဆောက်ပါမယ်။ EKS VPC တည်ဆောက်တဲ့ CloudFormation Template ကိုအောက်ကလင့်မှာ Clone or Download လုပ်လို.ရပါတယ်။ (**Official Link ကနေလဲ Download လုပ်လို.ရပါတယ်**)။

https://github.com/phyominhtun1990/AWS-EKS-Cluster-Introduction.git

Download လုပ်ပြီးပြီဆိုရင်ကျွန်တော်တို. တွေ CloudFormation Console ကိုသွားပါမယ်။ပြီးနောက် VPC Create template yml file ကို အောက်ဖော်ပြပါအတိုင်း upload template နေရာမှာ upload တင်ပါမယ်။



Upload Template in CF Console



View in designer in CF console

VPC မဆောက်ခင် တစ်ချက်ကြည့်ကြည့် ရင် VPC က Subnet သုံးခု အကြမ်းဖျဉ်းပါပါတယ်။ ထိုနောက်အောက်ဖော်ပြပါအတိုင်းဆက်လက်တည်ဆောက်ပါမယ်။ကျွန်တော်ကတော့ Template ထဲကအတိုင်း default ထားလိုက်ပြီးတည်ဆောက်ပါမယ်။

Stack name

Stack name

EKS-VPC

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

Worker Network Configuration

VpcBlock

The CIDR range for the VPC. This should be a valid private (RFC 1918) CIDR range.

192.168.0.0/16

Subnet01Block

CidrBlock for subnet 01 within the VPC

192.168.64.0/18

Subnet02Block

CidrBlock for subnet 02 within the VPC

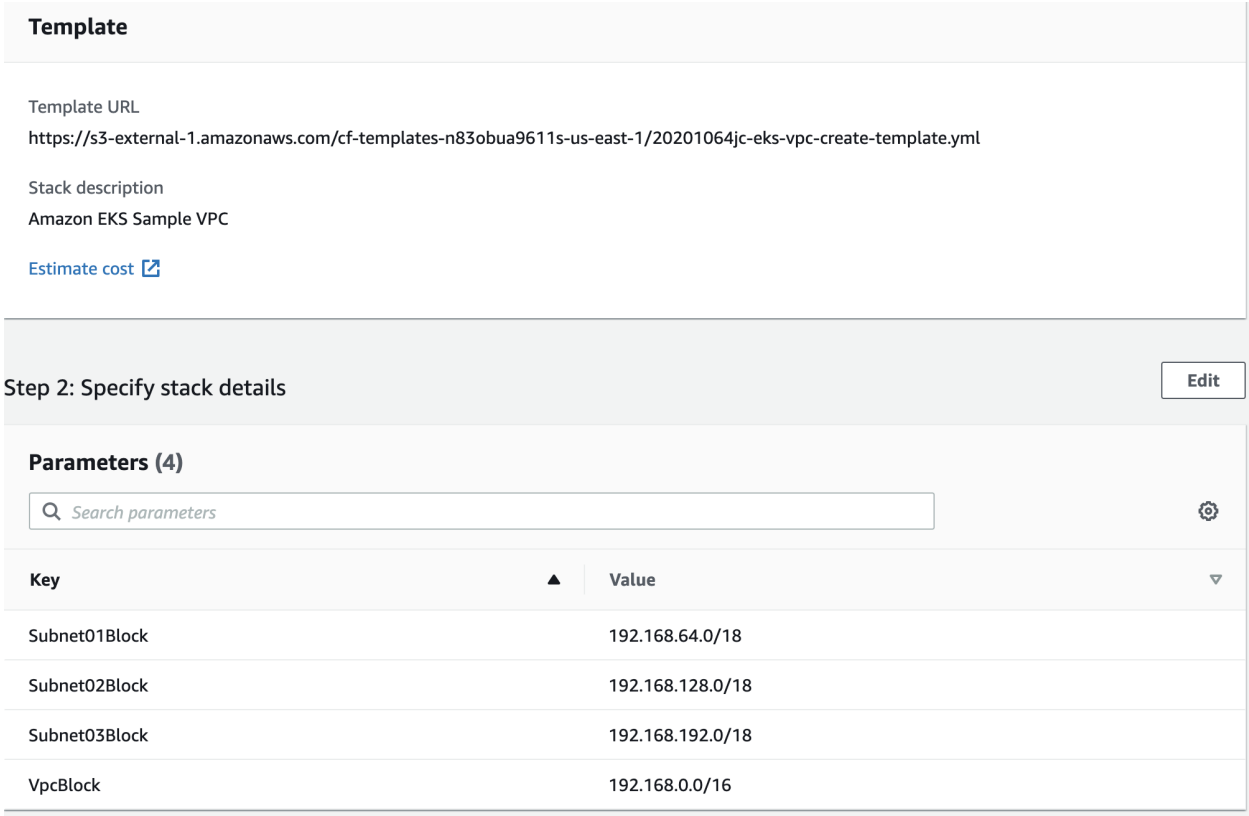
192.168.128.0/18

Subnet03Block

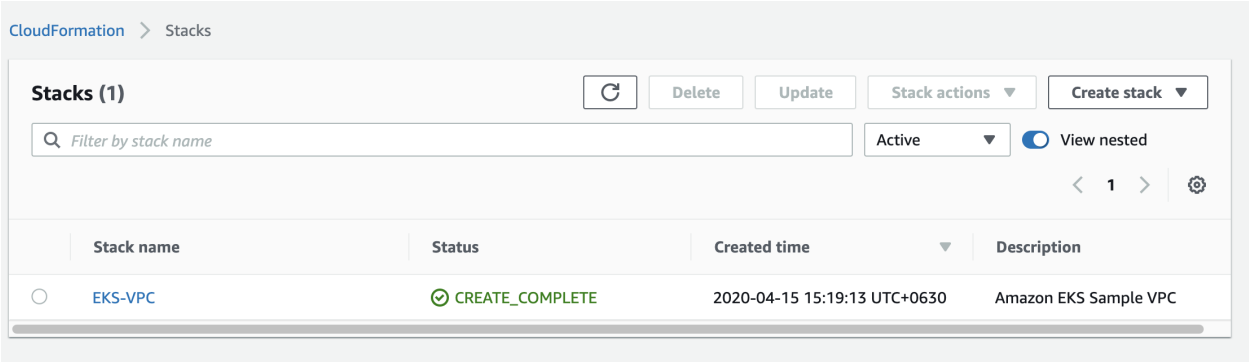
CidrBlock for subnet 03 within the VPC. This is used only if the region has more than 2 AZs.

192.168.192.0/18

VPC Network CIDR



VPC Network CIDR



EKS-VPC

ကျွန်တော်တို့. EKS Cluster တွက် VPC ကို CloudFormation Template နဲ့ တည်ဆောက်ပြီးသွားပြီဆိုတော့ ကျွန်တော်တို့.ဆက်ပြီး EKS Cluster ကို ဆက်လက်တည်ဆောက်ပါမယ်။ AWS EKS Cluster တည်ဆောက် ရာတွင် ကျွန်တော်တို့.သုံးမျိုး တည်ဆောက်လို့.ရပါတယ် AWS Management Console , AWS CLI နှင့် eksctl နဲ့တည်ဆောက်လို့.ရပါတယ်။ကျွန်တော်ကတော့ AWS CLI သုံးပြီး တော့တည်ဆောက်ပါမယ်။ အောက်မှာဖော်ပြထားတဲ့ Command (AWS CLI) နဲ့ EKS Cluster ကို တည်ဆောက်ပါမယ်။

AWS CLI ကို Configure လုပ်ပေးရပါမယ်။

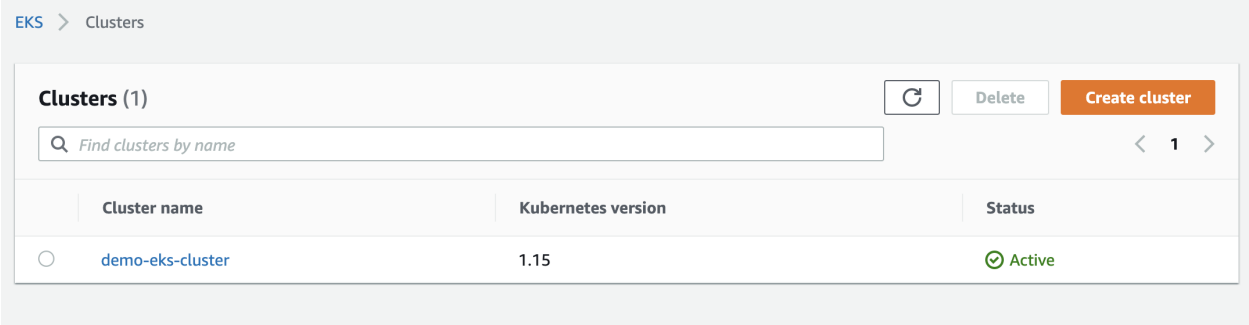
#aws configure

#aws eks --region region-code create-cluster --name demo-eks-cluster --role-arn arn:aws:iam::111122223333:role/EKS-Role --resources-vpc-config subnetIds=subnet-01,subnet-02,subnet-03,securityGroupIds=[sg-id]

```
phyominhtun@Phyos-MacBook-Pro-2:~$ aws eks --region us-east-1 create-cluster --name demo-eks-cluster --kubernetes-version 1.15 --role-arn arn:aws:iam::306837433561:role/EKS-Role --resources-vpc-config subnetIds=subnet-0ee5c12ba7f
7458,subnet-0595fa6c3ca5cd1c1,subnet-02d15cdbe87140b0c8,securityGroupIds=sg-02e712a6bceeef350
{
    "cluster": {
        "name": "demo-eks-cluster",
        "arn": "arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster",
        "createdAt": 1586942586.717,
        "version": "1.15",
        "roleArn": "arn:aws:iam::306837433561:role/EKS-Role",
        "resourcesVpcConfig": {
            "subnetIds": [
                "subnet-0ee5c12ba7f567458",
                "subnet-0595fa6c3ca5cd1c1",
                "subnet-02d15cdbe87140b0c8"
            ],
            "securityGroupIds": [
                "sg-02e712a6bceeef350"
            ],
            "vpcId": "vpc-0eecc28b9eed1220e",
            "endpointPublicAccess": true,
            "endpointPrivateAccess": false
        },
        "logging": {
            "clusterLogging": [
                {
                    "types": [
                        "api",
                        "audit",
                        "authenticator",
                        "controllerManager",
                        "scheduler"
                    ],
                    "enabled": false
                }
            ]
        },
        "status": "CREATING",
        "certificateAuthority": {},
        "platformVersion": "eks.2",
        "tags": {}
    }
}
phyominhtun@Phyos-MacBook-Pro-2:~$
```

EKS Cluster

EKS Cluster ကိုတည်ဆောက်ပြီးရင် ကျွန်တော်တို. EKS Console ကနေသွားကြည့်ရအောင်။ EKS Cluster ကိုတည်ဆောက်ပြီးရင်ကျွန်တော် တို. Console မှာ တွေ့.နိုင်ပါတယ်။



အိုခေ ကျွန်တော်တို. EKS Cluster တည်ဆောက် ပြီးပြီဆိုတော့ Worker Nodes တွေ မထည့်ခင် ကျွန်တော်တို. kube config ကို configure လုပ်ကြစို.။ EKS Cluster တွက် kube config ကို configure လုပ်ဖို.တွက်အောက်ဖော်ပြပါ command လေးနဲ့ configure လုပ်ကြရအောင်။

#aws eks --region us-east-1 update-kubeconfig --name demo-eks-cluster

```
phyominhtun@Phyos-MacBook-Pro-2:~/.kube$ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://35157AA5A46B7D221235C61E52313120.yl4.us-east-1.eks.amazonaws.com
  name: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
contexts:
- context:
    cluster: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
    user: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
  name: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
current-context: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
kind: Config
preferences: {}
users:
- name: arn:aws:eks:us-east-1:306837433561:cluster/demo-eks-cluster
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - --region
      - us-east-1
      - eks
      - get-token
      - --cluster-name
      - demo-eks-cluster
      command: aws
      env: null
phyominhtun@Phyos-MacBook-Pro-2:~/.kube$
```

Demo Cluster ဖြစ်တဲ့ အတွက် ပြီးရင် ပြန်ဖျက်မှာပါ။

demo-eks-cluster တွက် kube config လေး ထွက်လာပါလိမ့်မယ်။ config ရပြီဆိုတော့ကျွန်တော်တို့. kubectl command လေးနဲ့ နည်းနည်း ပါးပါးစမ်းကြည့်ရအောင်။

```
phyominhtun@Phyos-MacBook-Pro-2:~/.kube$ kubectl get ns,svc
NAME                          STATUS   AGE
namespace/default             Active   8h
namespace/kube-node-lease     Active   8h
namespace/kube-public         Active   8h
namespace/kube-system         Active   8h

NAME                 TYPE        CLUSTER-IP   EXTERNAL-IP   PORT(S)   AGE
service/kubernetes   ClusterIP   10.100.0.1   <none>        443/TCP   8h
phyominhtun@Phyos-MacBook-Pro-2:~/.kube$ kubectl get po -n kube-system
NAME                       READY   STATUS    RESTARTS   AGE
coredns-59dfd6b59f-f6hmg   0/1     Pending   0          8h
coredns-59dfd6b59f-svgjq   0/1     Pending   0          8h
phyominhtun@Phyos-MacBook-Pro-2:~/.kube$
```

ပုံပါအတိုင်းဆို ကျွန်တော်တို့. EKS cluster ကတော့ အလုပ်လုပ်နေပါပြီ။ဆိုတော့ကျွန်တော်တို့.တွေ Worker nodes တွေ ထည့်ကြည့်ရအောင်။

Worker nodes တွေထည့်ဖို့. ကျွန်တော်တို့. CloudFormation Template ကို အောက်က လင့်မှာ Download လုပ်လို့.ရပါတယ်။ (**Official Link ကနေလဲ Download လုပ်လို့.ရပါတယ်**)။

https://github.com/phyominhtun1990/AWS-EKS-Cluster-Introduction.git

Download လုပ်ပြီးပြီဆိုတော့ ကျနော်တို့. တွေ CloudFormation Template နဲ့ Worker nodes တွေကို တည်ဆောက်ကြရအောင်။ ပထမဆုံး Download လုပ်ထားတဲ့ CloudFormation Template လေးကို Upload လုပ်ပြီး Run လိုက်ပါမယ်။ Parameters တွေက တော့ အောက်ဖော်ပြပါပုံအတိုင်းထည့်ရပါမယ် တချို့. parameter တွေကို တော့ default ထားပါမယ်။

ami-0c5b63ec54dd3fc38 ကတော့ us-east-1 မှာဆိုရင်တော့ eks node ami ပါ။

Create stack

Prerequisite - Prepare template

Prepare template

Every stack is based on a template. A template is a JSON or YAML file that contains configuration information about the AWS resources you want to include in the stack.

- (●) Template is ready
- ( ) Use a sample template
- ( ) Create template in Designer

Specify template

A template is a JSON or YAML file that describes your stack's resources and properties.

Template source

Selecting a template generates an Amazon S3 URL where it will be stored.

- ( ) Amazon S3 URL
- (●) Upload a template file

Upload a template file

Choose file *eks-nodes-group-template.yml*

JSON or YAML formatted file

S3 URL: https://s3-external-1.amazonaws.com/cf-templates-n83obua9611s-us-east-1/20201066bk-eks-nodes-group-template.yml

View in Designer

Stack name

EKS-Worker-Node-Stack

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

EKS Cluster

ClusterName

The cluster name provided when the cluster was created. If it is incorrect, nodes will not be able to join the cluster.

demo-eks-cluster

ClusterControlPlaneSecurityGroup

The security group of the cluster control plane.

EKS-VPC-ControlPlaneSecurityGroup-L4UI7CU2GXDK (sg-02e712a6bceeef350)

Worker Node Configuration

NodeGroupName

Unique identifier for the Node Group.

EKS-Worker-Node

NodeAutoScalingGroupMinSize

Minimum size of Node Group ASG.

1

NodeAutoScalingGroupDesiredCapacity

Desired capacity of Node Group ASG.

3

NodeAutoScalingGroupMaxSize

Maximum size of Node Group ASG. Set to at least 1 greater than NodeAutoScalingGroupDesiredCapacity.

4

NodeInstanceType

EC2 instance type for the node instances

t3.medium

NodeImageId

AMI id for the node instances.

ami-0c5b63ec54dd3fc38

NodeVolumeSize

Node volume size

20

KeyName

The EC2 Key Pair to allow SSH access to the instances

phyominhtun

BootstrapArguments

Arguments to pass to the bootstrap script. See files/bootstrap.sh in https://github.com/awslabs/amazon-eks-ami

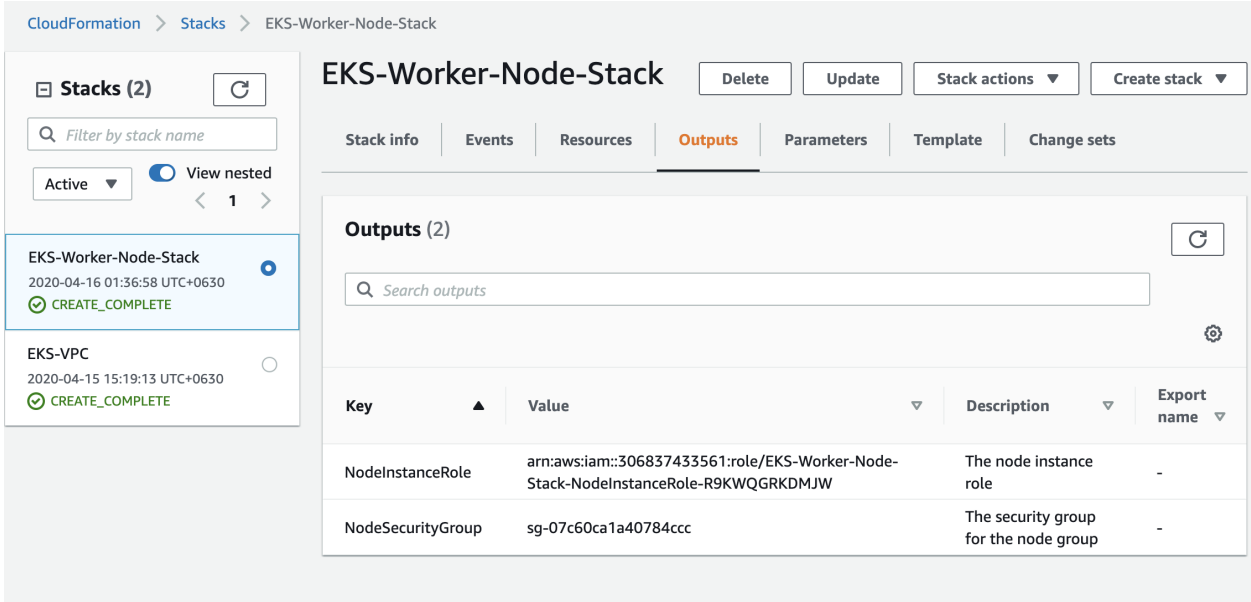
Worker Network Configuration

VpcId

The VPC of the worker instances

vpc-0eecc28b9eed1220e (192.168.0.0/16) (EKS-VPC-VPC)

Stack Complete ဖြစ်ပြီဆိုရင်ကျွန်တော်တို့. EKS Cluster နဲ့ Worker Nodes တွေကို Join ဖို့. တွက်အောက်ဖော်ပြပါ ပုံထဲက "**NodeInstanceRole**" ARN ကို မှတ်ထားရပါမယ်။



ထို.နောက် ကျွန်တော်တို့. Worker nodes ချိတ်ဆက်ဖို့.တွက် Configmap ကို Download လုပ်ပြီး cluster မှာ run ပေးရန်လိုအပ်ပါသည်။ အရင်ဆုံး auth Configmap ကို download

https://github.com/phyominhtun1990/AWS-EKS-Cluster-Introduction.git

ပြီးရင်ကျွန်တော်တို့. အပေါ်မှာ မှတ်ထားတဲ့ "**NodeInstanceRole**" ARN ကိုအစားထိုးပြီး run ပေးရပါမယ်။

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: aws-auth
  namespace: kube-system
data:
  mapRoles: |
    - rolearn: arn:aws:iam::306837433561:role/EKS-Worker-Node-Stack-NodeInstanceRole-R9KWQGRKDMJW
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
~
~
~
~
~
~
```

#kubectl apply -f aws-authentation-cm-workers.yaml

kubectl apply command နဲ့ Configmap ကို run ပြီးရင် တခဏ လောက်ကြာရင် #kubectl get nodes command ရိုက်လိုက်ရင် အောက်ဖော်ပြပါပုံအတိုင်း worker nodes တွေ Ready ဖြစ်နေတာကိုတွေ့.ရပါလိမ့်မည်။

```
phyominhtun@Phyos-MacBook-Pro-2:~/Documents/AWS-EKS-Cluste
NAME                                        STATUS   ROLES    AGE
ip-192-168-123-11.ec2.internal              Ready    <none>   30m
ip-192-168-132-196.ec2.internal             Ready    <none>   30m
ip-192-168-242-102.ec2.internal             Ready    <none>   30m
```

```
ip-192-168-242-102.ec2.internal   Ready    <none>   30m   v1.11.5
phyominhtun@Phyos-MacBook-Pro-2:~/Documents/AWS-EKS-Cluster-Introduction$ kubectl get po -n kube-system
NAME                      READY   STATUS    RESTARTS   AGE
aws-node-9t984            1/1     Running   0          33m
aws-node-bn2dj            1/1     Running   0          33m
aws-node-pvrzc            1/1     Running   0          33m
coredns-59dfd6b59f-f6hmg  1/1     Running   0          10h
coredns-59dfd6b59f-svgjq  1/1     Running   0          10h
kube-proxy-bbk9p          1/1     Running   0          33m
kube-proxy-g2cp5          1/1     Running   0          33m
kube-proxy-q92r4          1/1     Running   0          33m
```

ထိုအပြင် kube-system namespace ထဲ က pods တွေလည်း running ဖြစ်နေတာကိုတွေ့.မြင်ရပါလိမ့်မယ်။အိုခေ ,ဒါဆို Node လည်း Ready ဆို ကျန်တာတွေလည်း running ဆိုတော့ EKS Cluster က up and running ဖြစ်နေပြီပေါ့ဗျာ။ သေချာအောင်ကျွန်တော် တို. Application တစ်ခုလောက် run ကြည့်ကြရအောင်ဗျာ။ လွယ်လွယ်ကူကူလေးဖြစ်အောင် 2048 game လေးကို EKS Cluster မှာ run ကြည့်ရအောင်။

2048 yaml file လေးကို အောက်က လင့်လေးမှာ Download လုပ်လို.ရပါတယ်။ Deployment file and service file လေးနှစ်ခုကို Download လုပ်ပြီး terminal ကနေ အောက်ပါ command လေးရိုက်ပြီး run နိုင်ပါတယ်။

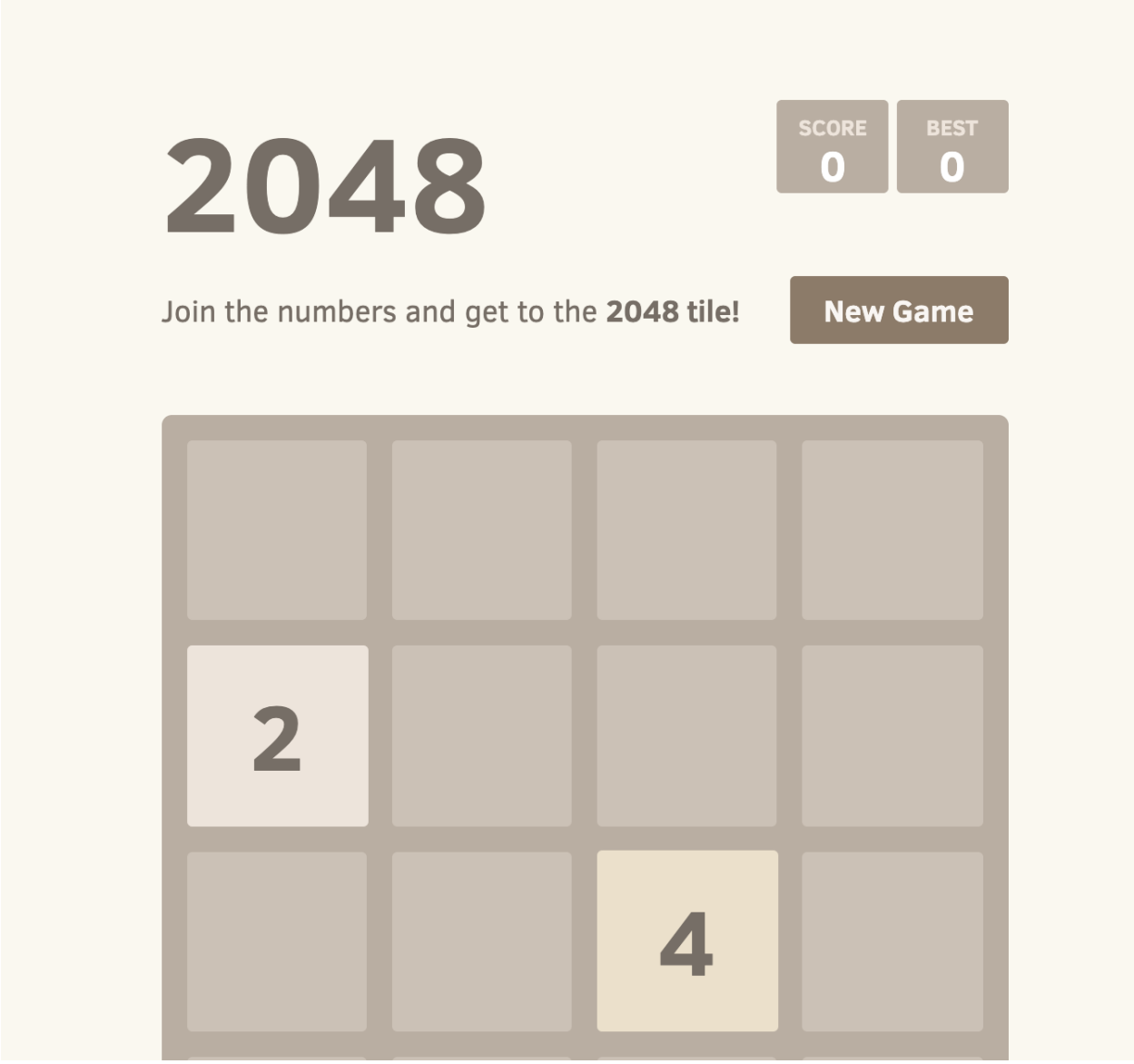
#kubectl apply -f file.yml

https://github.com/phyominhtun1990/AWS-EKS-Cluster-Introduction.git

```
phyominhtun@Phyos-MacBook-Pro-2:~/Documents/AWS-EKS-Cluster-Introduction/2048-game$ kubectl get deploy,svc -n 2048-game
NAME                                  READY   UP-TO-DATE   AVAILABLE   AGE
deployment.extensions/2048-deployment 3/3     3            3           4m24s

NAME                  TYPE       CLUSTER-IP     EXTERNAL-IP   PORT(S)        AGE
service/service-2048  NodePort   10.100.36.23   <none>        80:31957/TCP   3m18s
phyominhtun@Phyos-MacBook-Pro-2:~/Documents/AWS-EKS-Cluster-Introduction/2048-game$
```

အိုခေ ကျွန်တော်တို. Cluster မှာ namespace ဆောက်တယ် deployment ကို replica သုံးလုံး ဆောက်တယ်ပြီးတော့ NodePort service ထုတ်လိုက်တယ်။ ဒါဆို NodePort (31957)ကို instance security group မှာ allow လုပ်ပြီး 2048 game ကို access လုပ်ကြည်ကြရအောင်။

http://34.239.94.225:31957/



အိုခေပြီဗျာ .. ကျွန်တော်တို. AWS EKS Cluster က up and running လည်းဖြစ် application ပါတင်ပြီး Run လို.ရနေပါပြီ။ ဒါဆို ကျွန်တော်တို. ဒီ tutorial လေးကို မိတ်ဆွေတို. ကိုယ်တိုင် လိုက်လုပ်ရင်း EKS Cluster ရဲ. feature တွေ kubernetes cluster ရဲ. feature တွေကို ကိုယ်ကိုတိုင် လိုက်လုပ်ကြည့်ရင်း ကိုယ်ပိုင် cluster များ ဆောက်နိုင်ပါစေဟုပြောကြားရင်း နိဂုံးချုပ်ပါရစေ။

ကျွန်တော်တို့. AWS User Group Myanmar က ပြုလုပ်မဲ့ Online Meetup မှာ ပြန်ဆုံပြီး ထပ်ပြီးဆွေးနွေးကြမယ်။

ကျေးဇူးတင်လျက် ... AWS User Group Myanmar

Stay Safe and Healthy!

Infrastructure as Code with AWS CodePipeline

We Love IaC 3000! (AWS CloudFormation, Terraform and AWS CodePipeline)

Infrastructure as Code with AWS CodePipeline

AWS User Group Myanmar Blog မှာ tutorial တစ်ခုရေးမယ်ဆိုပြီး အလုပ်မအားတာက တကြောင်း .. Project တွေပိနေတာနဲ့ မရေးဖြစ်တာတောင်တော်တော်ကြာသွားပြီ နဂို စာရေးပျင်းတာလည်းပါတာပေါ့။ အခု Project တစ်ခုလုပ်နေရင်း Idea လေးတစ်ခုရတာနဲ့ Infrastructure as Code ကို AWS မှာ Continuous Delivery Pipeline နဲ့လုပ်တာလေးရေးမယ်ဆိုပြီး ဒီ tutorial လေးကိုရေးဖြစ်တာ။ ဆိုတော့ဒီ tutorial လေးကို အစက AWS User Group Myanmar Meetup တွေ Workshop event တွေမှာပြောမယ်ပေါ့.. Covid19 ဖြစ်သွားတော့ပွဲလည်းအပြင်မှာ မလုပ်ဖြစ်တော့ Online ပွဲမှာ Workshop လုပ်ရတာအားလည်းမရတာနဲ့ စာပဲရေးမယ်ဆိုပြီးဆုံးဖြတ်ဖြစ်သွားတယ်။ ဒီ tutorial မှာ နှစ်ပိုင်းပါမယ် ။ တစ်ပိုင်းက ကျနော် IaC Pipeline ကို AWS CloudFormation and AWS CodePipeline နဲ့.အသုံးပြုတာကိုရေးသွားမယ် နောက်တစ်ပိုင်းက ညီလေး ဒီသာထွန်း က သူက Terraform နဲ့ ရေးထားတာရှိတယ်ဆိုတာနဲ့ နှစ်ယောက်ပေါင်းပြီးရေးဖြစ်သွားတယ်ပေါ့။ Terraform ကို AWS CodePipeline နဲ့ AWS CloudFormation ကို AWS CodePipeline နဲ့ဆိုပြီးတော့နှစ်ပိုင်းပေါ့။

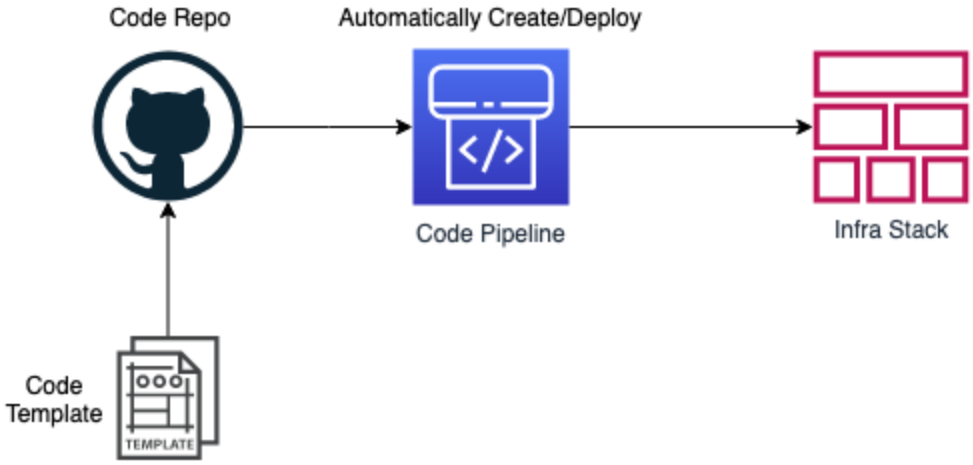
အိုခေ ဒါဆိုနိဒါန်း ချေနေတာကလည်း တော်တော်များပြီဆိုတော့ ပထမဆုံးအနေနဲ့ **Infrastructure as Code (IaC) ဆိုတာဘာလဲ?** ဘာအတွက်သုံးတာလဲ? ဆိုပြီးမေးခွန်းတွေရှိတယ်။ ဆိုတော့ကာ နောက်ပိုင်း DevOps တို့. Automation တွေ အလုပ်များလာတာနဲ့အမျှ Infrastructure as Code ကိုနောက်ပိုင်း Organization Infrastructure တွေမှာ အသုံးပြုလာကြပါတယ်။ အရှင်းဆုံးပြောရရင်တော့

Infrastructure as Code ဆိုတာ လက်ရှိကျနော်တို.တွေ manual တည်ဆောက်ကြတဲ့ Infrastructure တွေ Server, Database အစရှိတဲ့ resource တွေကို code နဲ့ automated တည်ဆောက်တာကို ဆိုလိုပါတယ် Cloud Infrastructure မှာဖြစ်ဖြစ် On-Premesis Infrastructure မှာပဲ ဖြစ်ဖြစ်ပေါ့။
Code နဲ့တည်ဆောက်တယ်ဆိုတာ Programming Code တစ်ခုခု နဲ့ဖြစ်ဖြစ် Configuration Code တွေနဲ့ဖြစ်ဖြစ် တည်ဆောက်နိုင်ပါတယ်။ အထူးသဖြင့် infrastructure orchestration tools တွေဖြစ်တဲ့ Terraform တို. AWS မှာ ဆို AWS CloudFormation တို.ကို အသုံးပြုလေ့ရှိပါတယ်။တခြား Provider တွေက tools တွေလည်းရှိပါတယ်။သေချာလေ့လာပြီး သိချင်ရင်အောက်က လင့်မှာဖတ်နိုင်ပါတယ်။

https://blog.newrelic.com/engineering/best-cloud-infrastructure-automation-tools/

အဲ့တော့ Infrastructure ကို Code တွေနဲ့ရေးပြီးတည်ဆောက်မယ် ပြီးတော့ Code Repository ဖြစ်တဲ့ Github မှာတင်လိုက်တာနဲ့ Auto Deploy သွားနိုင်အောင် CI/CD Pipeline တွက် AWS Code Pipeline ကိုအသုံးပြုပြီး Deploy သွားမယ်။

https://aws.amazon.com/codepipeline/



ကျွန်တော်တိုအပေါ်က Diagram လို Infrastructure as Code တွက် AWS CloudFormation Template ရေးမယ် ပြီးရင် GitHub Repository မှာ Push မယ် Push လိုက်တာနဲ့ တပြိုင်နက် AWS CodePipeline က Trigger မိပြီး Automatically Infrastructure Stacks တွေကို သွားဆောက်မယ် အဲ့လိုဆိုတော့ ကျနော်တို.အရင် ဆုံး AWS CloudFormation Template ရေးဖို.တွက်ကို ကျနော် စလုပ်ပါမယ်။ဒီ tutorial မှာ စမ်းမှာက Amazon S3 Static WebHost Bucket ကို CloudFormation နဲ. auto တည်ဆောက်တဲ့

Template ကို သုံးမှာဖြစ်ပြီး ကိုယ်ကိုတိုင်အသုံးပြုရင်တော့ ကိုယ်ကြိုက်တဲ့ resource ကို CloudFormation နဲ့တည်ဆောက်လို့.ရပါတယ် ဥပမာ EC2 တို့. RDS တို့. VPC တို့.ဆောက်တာပေါ ့။

အိုခေ ဒါဆို ပထမဆုံး အနေနဲ့ ကျနော်တို့. CloudFormation မလုပ်ခင်အရင်ဆုံး ကျွန်တော်တို့. AWS IAM Role တွေတည်ဆောက်ရပါမယ်။ Role ကတော့ **နှစ်ခု** ဆောက်ပေးရပါ။

1. CodePipeline IAM Role
2. CloudFormation IAM Role ဆိုပြီးတော့ Role နှစ်ခုအရင်ဆောက်ရပါမယ်။ ဒီ လင့်ကနေ policy ကို download လုပ်ပြီးတည်ဆောက်နိုင်ပါတယ်။

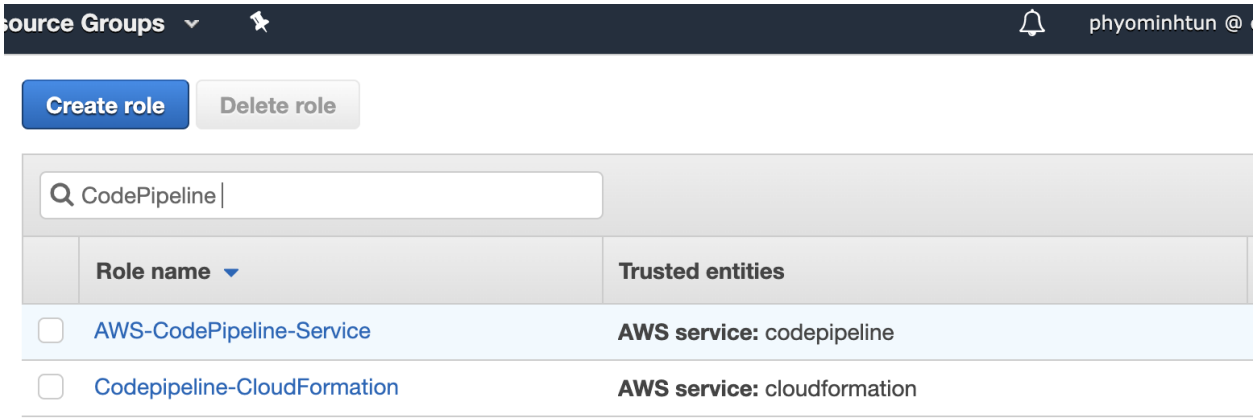
https://github.com/phyominhtun1990/cloudformation-codepipeline

```
{
  "Statement": [
    {
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:GetBucketVersioning"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::codepipeline*",
        "arn:aws:s3:::elasticbeanstalk*"
      ],
      "Effect": "Allow"
    },
    {
      "Action": [
        "codecommit:CancelUploadArchive",
        "codecommit:GetBranch",
        "codecommit:GetCommit",
        "codecommit:GetUploadArchiveStatus",
        "codecommit:UploadArchive"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
```

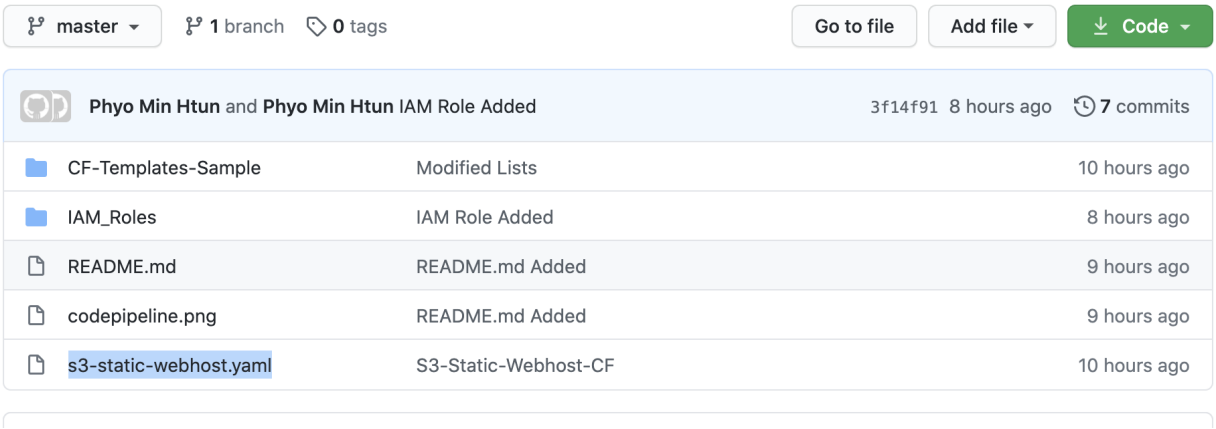
```
{
    "Action": [
        "codedeploy:CreateDeployment",
        "codedeploy:GetApplicationRevision",
        "codedeploy:GetDeployment",
        "codedeploy:GetDeploymentConfig",
        "codedeploy:RegisterApplicationRevision"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
{
    "Action": [
        "elasticbeanstalk:*",
        "ec2:*",
        "elasticloadbalancing:*",
        "autoscaling:*",
        "cloudwatch:*",
        "s3:*",
        "sns:*",
        "cloudformation:*",
        "rds:*",
        "sqs:*",
        "ecs:*",
        "iam:PassRole"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
{
    "Action": [
        "lambda:InvokeFunction",
        "lambda:ListFunctions"
    ],
    "Resource": "*",
    "Effect": "Allow"
},
{
    "Action": [
        "opsworks:CreateDeployment",
        "opsworks:DescribeApps",
        "opsworks:DescribeCommands",
        "opsworks:DescribeDeployments",
```

```
            "opsworks:DescribeInstances",
            "opsworks:DescribeStacks",
            "opsworks:UpdateApp",
            "opsworks:UpdateStack"
        ],
        "Resource": "*",
        "Effect": "Allow"
    },
    {
        "Action": [
            "cloudformation:CreateStack",
            "cloudformation:DeleteStack",
            "cloudformation:DescribeStacks",
            "cloudformation:UpdateStack",
            "cloudformation:CreateChangeSet",
            "cloudformation:DeleteChangeSet",
            "cloudformation:DescribeChangeSet",
            "cloudformation:ExecuteChangeSet",
            "cloudformation:SetStackPolicy",
            "cloudformation:ValidateTemplate",
            "iam:PassRole"
        ],
        "Resource": "*",
        "Effect": "Allow"
    },
    {
        "Action": [
            "codebuild:BatchGetBuilds",
            "codebuild:StartBuild"
        ],
        "Resource": "*",
        "Effect": "Allow"
    },
    {
        "Effect": "Allow",
        "Action": [
            "devicefarm:ListProjects",
            "devicefarm:ListDevicePools",
            "devicefarm:GetRun",
            "devicefarm:GetUpload",
            "devicefarm:CreateUpload",
            "devicefarm:ScheduleRun"
        ],
```

```
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "servicecatalog:ListProvisioningArtifacts",
                "servicecatalog:CreateProvisioningArtifact",
                "servicecatalog:DescribeProvisioningArtifact",
                "servicecatalog:DeleteProvisioningArtifact",
                "servicecatalog:UpdateProduct"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudformation:ValidateTemplate"
            ],
            "Resource": "*"
        }
    ],
    "Version": "2012-10-17"
}
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Sid": "",
     "Effect": "Allow",
     "Principal": {
       "Service": "codepipeline.amazonaws.com"
     },
     "Action": "sts:AssumeRole"
   }
 ]
}
```



ပြီးရင်အပေါ်ကပုံအတိုင်းကျနော်တို့. IAM Role နှစ်ခုရလာပါလိမ့်မယ်။ပြီးသွားရင်ကျနော်တို့.တွေ S3 Static WebHost Bucket တည်ဆောက်တဲ့ AWS CloudFormation Template ကို Github ပေါ်ကိုတင်ပါမယ်။ကျနော်ကတော့တင်ထားပြီးသားဆိုတော့ GitHub ပေါ်မှာရောက်နေပါလိမ့်မယ်။

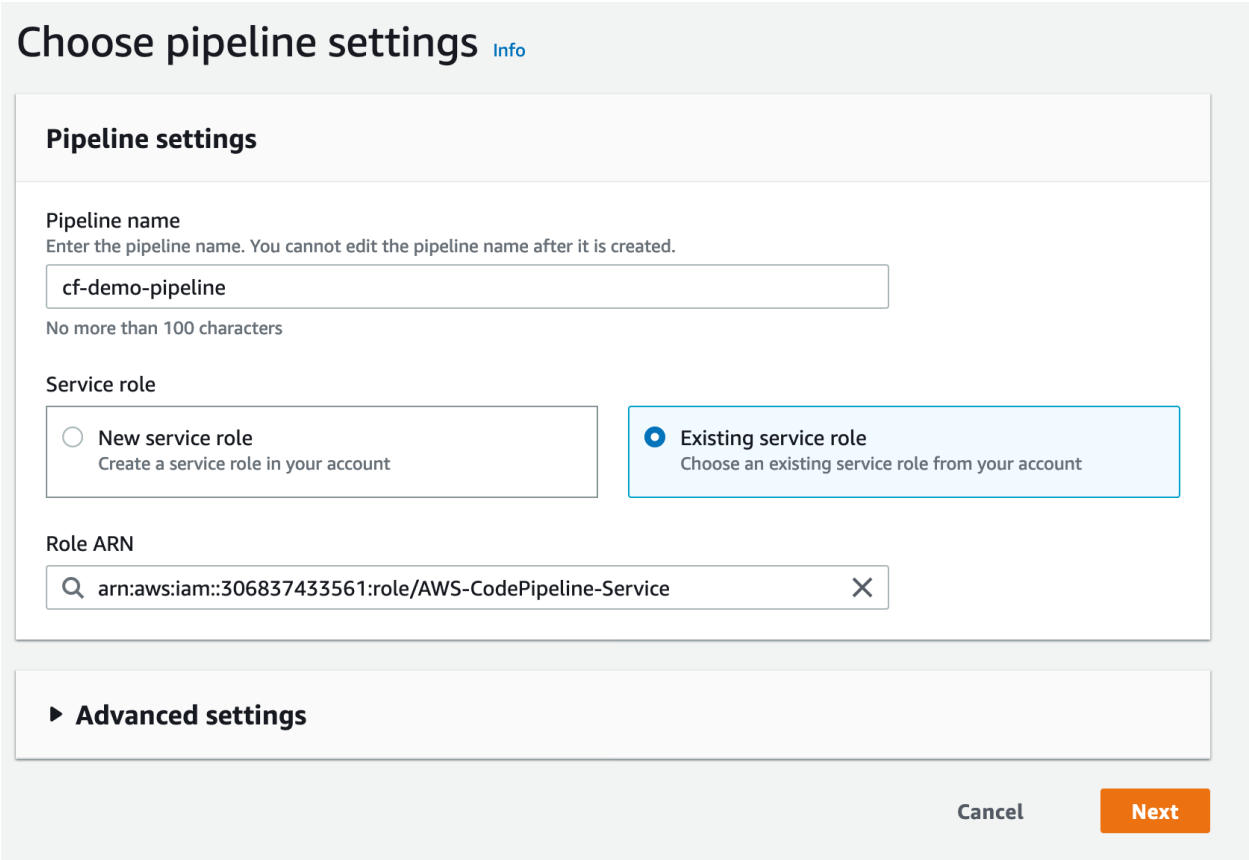



```
## CloudFormation Template Sample - AWS S3 StaticWebHost Bucket

AWSTemplateFormatVersion: 2010-09-09
Resources:
 S3Bucket:
   Type: AWS::S3::Bucket
   Properties:
     AccessControl: PublicRead
     WebsiteConfiguration:
       IndexDocument: index.html
       ErrorDocument: error.html
 BucketPolicy:
   Type: AWS::S3::BucketPolicy
   Properties:
     PolicyDocument:
       Id: Mys3bucketPolicy
```

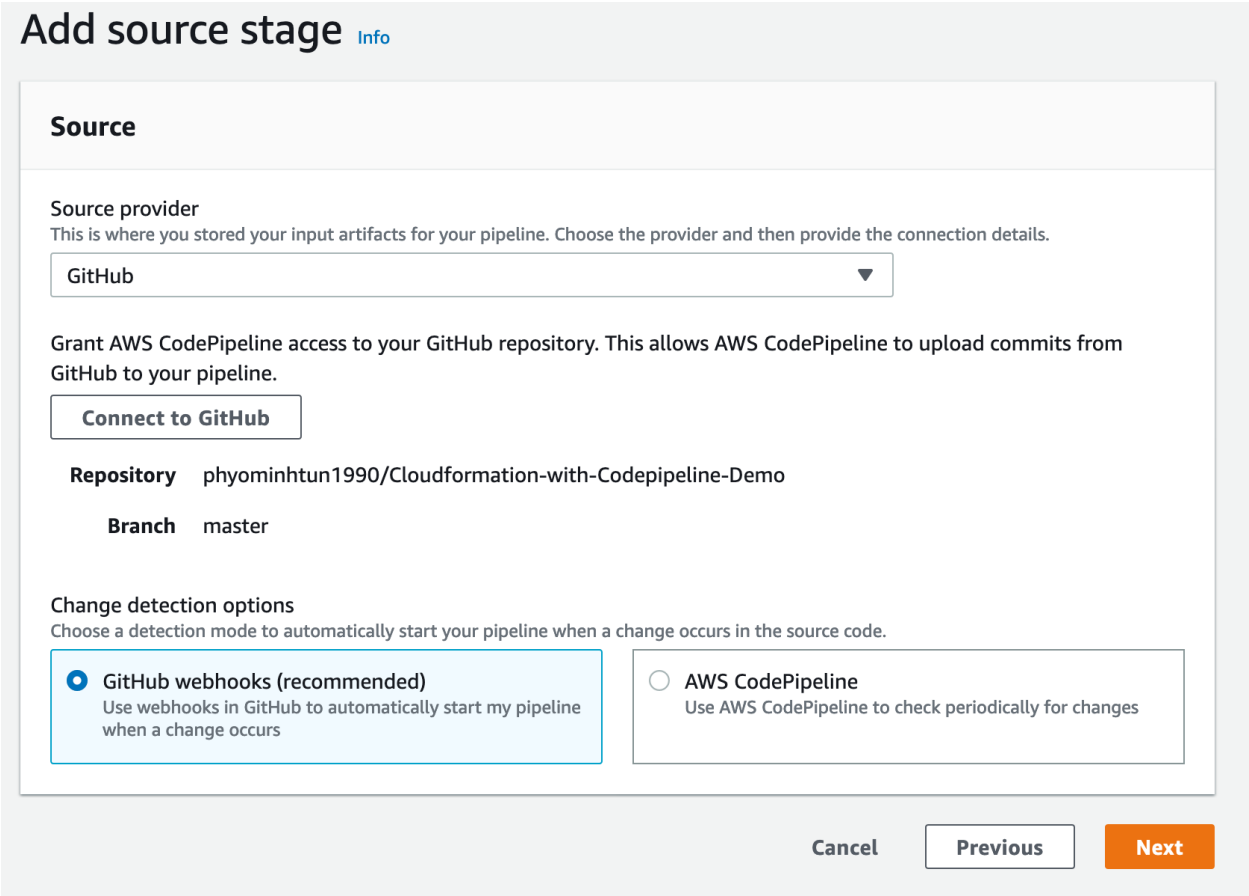
```
      Version: 2012-10-17
      Statement:
        - Sid: PublicReadForGetBucketObjects
          Effect: Allow
          Principal: '*'
          Action: 's3:GetObject'
          Resource: !Join
            - ''
            - - 'arn:aws:s3:::'
              - !Ref S3Bucket
              - /*
    Bucket: !Ref S3Bucket
Outputs:
 WebsiteURL:
   Value: !GetAtt
     - S3Bucket
     - WebsiteURL
   Description: URL for website hosted on S3
 S3BucketSecureURL:
   Value: !Join
     - ''
     - - 'https://'
       - !GetAtt
         - S3Bucket
         - DomainName
   Description: Name of S3 bucket to hold website content
```

ဒီလိုရောက်နေပြီဆိုရင် ကျွန်တော်တို. AWS Management Console ကနေ AWS CodePipeline ကိုတည်ဆောက်ပါမယ်။ ပြီးရင်ကျနော်တို. AWS CodePipeline နဲ့ Github နဲ့ကို ချိတ်ပါမယ်။ အရင်ဆုံး AWS Console ကနေ CodePipeline Service ကိုသွားမယ်။ ပြီးရင် Create Pipeline ဆိုပြီး Pipeline အသစ်တစ်ခုဆောက်မယ်။

အဲ့ဒီမှာ Role နေရာမှာ Existing service role ကိုရွေးချယ်ပြီး အပေါ်မှာတည်ဆောက်ထားတဲ့ IAM Role နှစ်ခုထဲက CodePipeline IAM Service Role ကို ရွေးထည့်ပေးရပါမယ်။



ပြီးသွားရင် Add Source Stage မှာ GitHub Repo နဲ့ချိတ်ပေးရပါမယ်။ချိတ်ဆက်ပြီးသွားရင်တော့ ကိုယ့် Github Repo တွေကိုတွေ့.ရပါမယ်။ အဲ့ကနေမှာ ခုနက ကိုယ် CloudFormation Template store ထားတဲ့ Repo ကို ရွေးချယ်ပြီး Webhooks တည်ဆောက်ပေးရပါမယ်။



ပြီးရင် Build Stage ကို ကျော်မယ်။ ထို.နောက် Deployment Provider ကို AWS CloudFormation ရွေးပေးပြီး Artifact name မှာ SourceArtifact နှင့် File name မှာ AWS CloudFormation Template name ကို ထည့်ပေးရပါမယ်။ Role Name မှာတော့ အပေါ်ကတည်ဆောက်ခဲ့သော IAM Role နှစ်ခုထဲက CloudFormation Role ကို ရွေးချယ်ပေးရပါမယ်။

Deploy provider
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

AWS CloudFormation

Region

Asia Pacific (Singapore)

Action mode
When you update an existing stack, the update is permanent. When you use a change set, the result provides a diff of the updated stack and the original stack before you choose to execute the change.

Create or update a stack

Stack name
If you are updating an existing stack, choose the stack name.

s3-static-webhost-creation-codepipeline

Template
Specify the template you uploaded to your source location.

Artifact name	File name	Template file path
SourceArtifact	s3-static-webhost.yaml	SourceArtifact::s3-static-

Template configuration - *optional*
Specify the configuration file you uploaded to your source location.

Use configuration file

Artifact name	File name	Template configuration file path

Capabilities - *optional*
Specify whether you want to allow AWS CloudFormation to create IAM resources on your behalf.

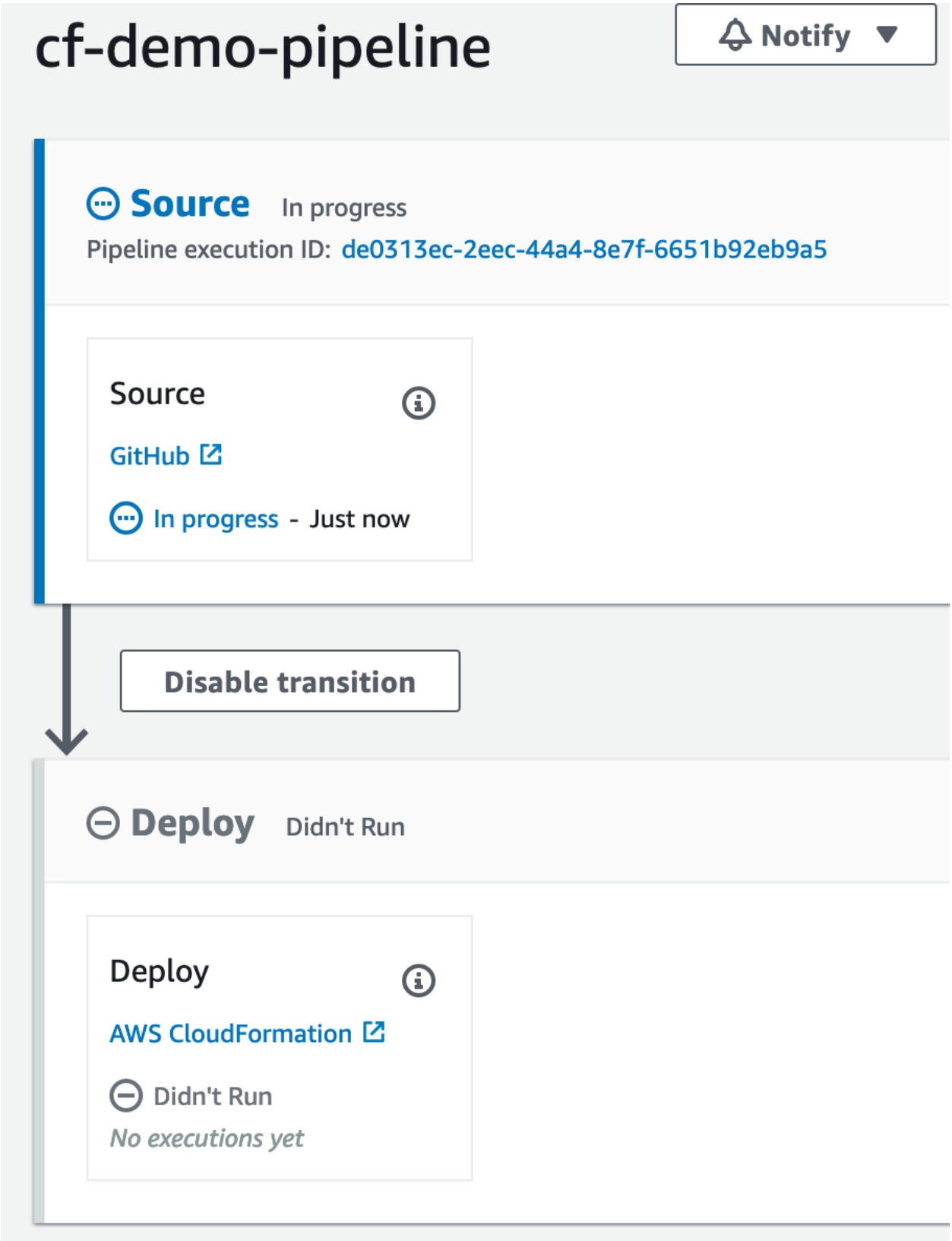
Role name

arn:aws:iam::306837433561:role/Codepipeline-CloudFormation

Output file name

File generated by this action

ပြီးတာနဲ့တပြိုင်နက်ထဲ AWS CodePipeline က ချက်ချင်းကို အောက်ကပုံပါအတိုင်း စပြီး Source Repo ကနေဆွဲချပြီး Deployment Process ကို စတင်ပါတော့တယ်။


cf-demo-pipeline
Notify
Source
In progress
Pipeline execution ID: de0313ec-2eec-44a4-8e7f-6651b92eb9a5
Source
GitHub
In progress - Just now
Disable transition
Deploy
Didn't Run
Deploy
AWS CloudFormation
Didn't Run
No executions yet

Source Succeeded

Pipeline execution ID: de0313ec-2eec-44a4-8e7f-6651b92eb9a5

Source

GitHub

Succeeded

- 1 minute ago

18de0ccf

18de0ccf Source: S3-Static-Webhost-CF

Disable transition

Deploy Succeeded

Pipeline execution ID: de0313ec-2eec-44a4-8e7f-6651b92eb9a5

Deploy

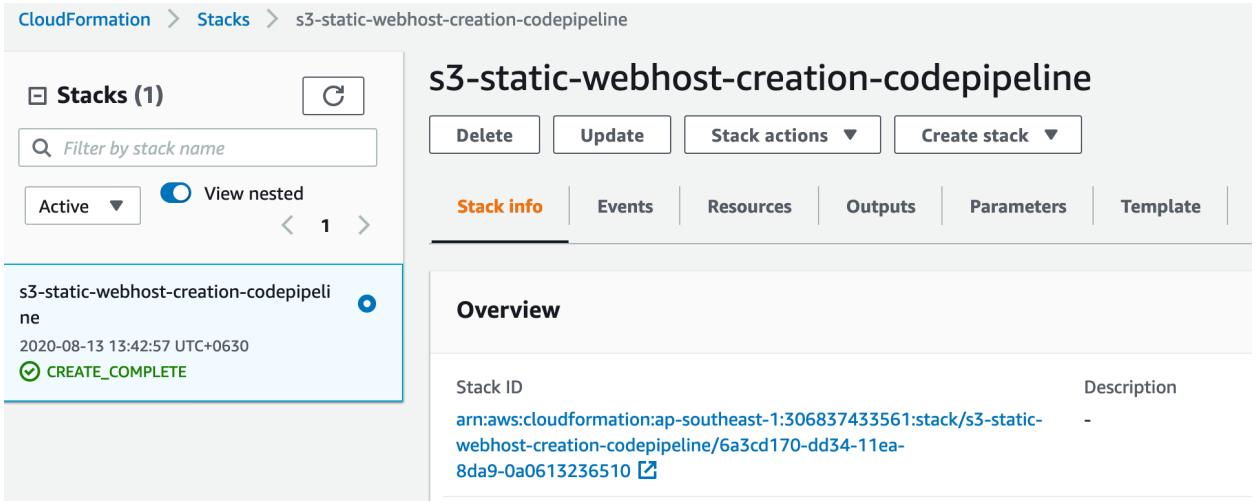
AWS CloudFormation

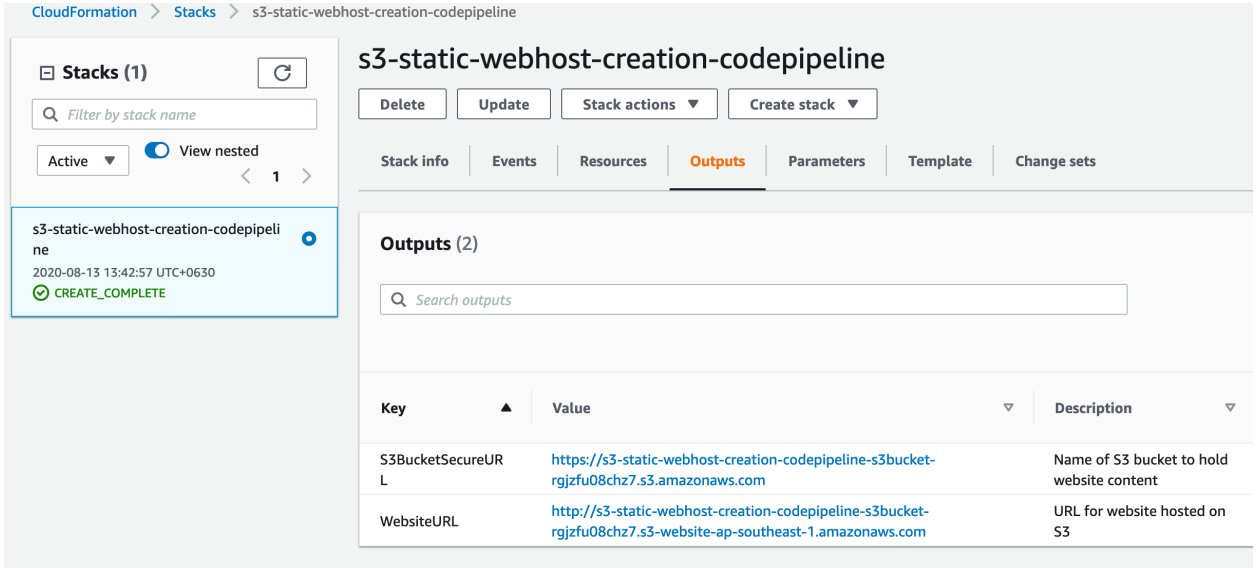
Succeeded - Just now

Details

18de0ccf Source: S3-Static-Webhost-CF

အပေါ်ကပုံအတိုင်း အားလုံး အိုကေတယ်ဆိုရင် Deployment Success ဖြစ်သွားတာပေါ့။ ဒါဆိုရင်ကျွန်တော်တို.တွေ AWS CloudFormation Console ကိုသွားပြီး ကြည့်ရအောင်။





အိုကေ...ဒါဆိုကျွန်တော်တို. CodePipeline ကနေ CloudFormation ကို Deploy လုပ်ပြီးတော့ S3 Static WebHost လုပ်ဖို.အတွက် Bucket ကို CloudFormation Template နဲ့ တည်ဆောက်ပြီးဖြစ်ပါတယ်။

s3-static-webhost-creation-cod...

Copy Bucket ARN

Properties

Events	0 Active notifications
Versioning	Disabled
MFA delete	Disabled
Logging	Disabled
Static web hosting	Enable website hosting
Tags	3 Tags
Requester pays	Disabled
Object lock	Disabled
Transfer acceleration	Disabled

Permissions

Owner	phyominhtun.opslab
Block public access	Disabled
Bucket policy	Yes
Access control list	2 Grantees
CORS configuration	No

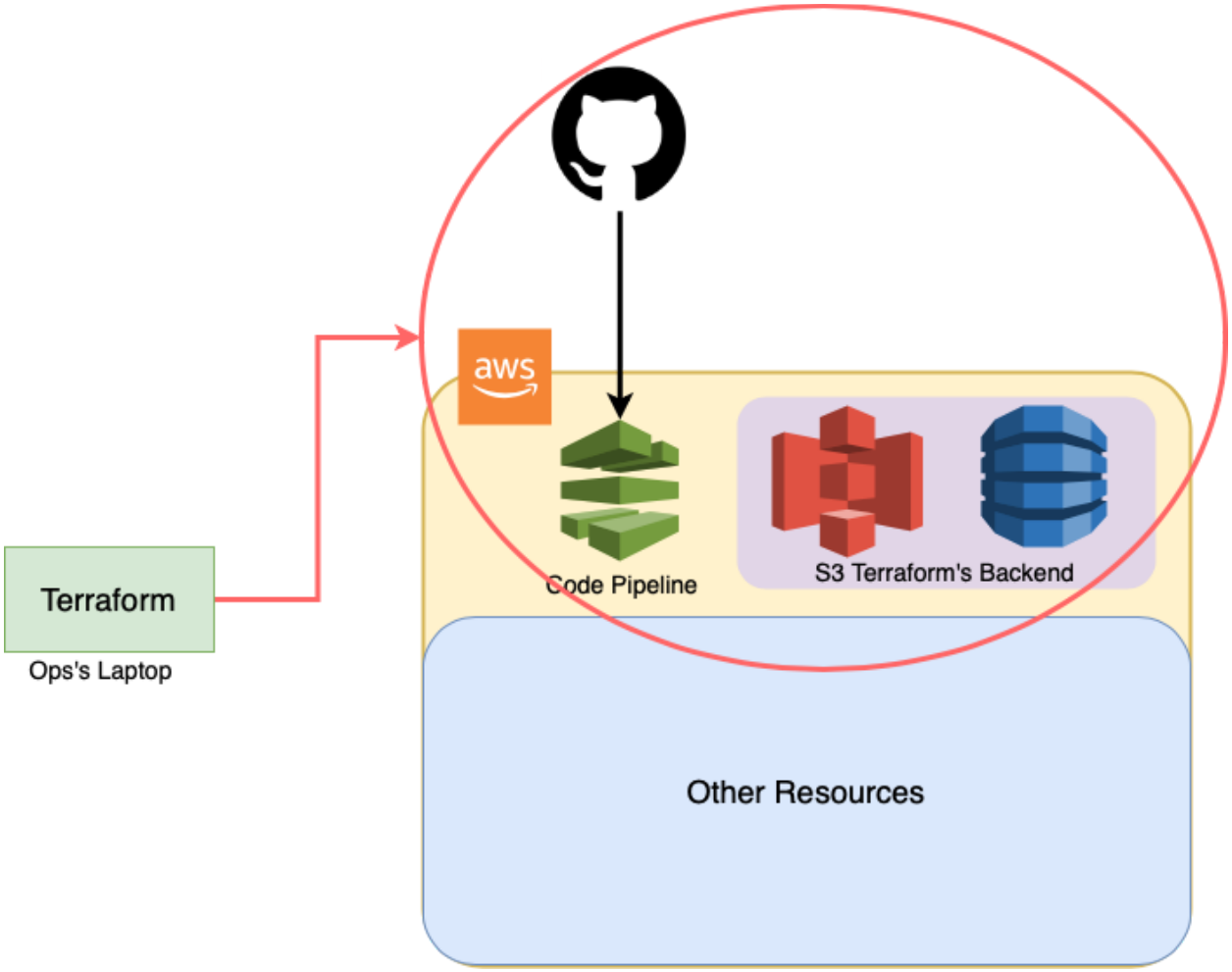
Management

Lifecycle	Disabled
Replication	Disabled
Analytics	Disabled

ဒါဆိုကျွန်တော်တို. CodePipeline ကနေဆောက်ထားတဲ့ Bucket မှာ Web data တင်ပြီးတော့ Static Website တစ်ခု run လို.ရပါပြီ။ နောက်တစ်ပိုင်းကတော့ ကျွန်တော် ညီလေး ဒီသာထွန်းရဲ့ Terraform နဲ့ Pipeline တည်ဆောက်တာကိုဆက်ရေးသွားမှာဖြစ်ပါတယ်။ အောက်တွင်ဆက်လက်ဖတ်ရှုပေးကြပါရန် ။။

IAC Using Terraform With AWS Code Pipeline

အခုကတော့ အပေါ်မှာ ပြောခဲ့ တဲ့ ပုံစံမျိုး အလားတူ ကို Terraform အသုံးပြုပြီး Terraform fan တွေအတွက် ဖော်ပြပေးသွားမှာ ဖြစ်ပါတယ်။။ Infrastructure ကို code အနေနဲ့ စတင် အသုံးပြုမယ်၊ Infrastrure ကိုလဲ အရည်သွေးကောင်းမွန်အောင်၊ စနစ်တစ်ကျဖြစ်အောင်၊ Application/Software တွေကို Treat သလိုမျိုး Treat သင့်ပါတယ်။ ဒါဆိုရင်တော့ Build/Test အမျိုးမျိုး ပြုလုပ်နိုင်ဖို့ CI/CD Pipeline နဲ့ Environment တွေ ဖန်တီးဖို့ လိုအပ်လာမှာ ဖြစ်ပါတယ်။ အခုအကြောင်းအရာမှာတော့ Infra ကို Environment တွေခွဲပြီး Testing တွေ integration တွေ ပြုလုပ်ဖို့ထက် basic code pipeline အရင် အလွယ်တကူတည် ဆောက်နိုင်ဖို့ ပြောပြပေးသွားမှာ ဖြစ်ပါတယ်။ အရင်ဦးဆုံး ဘာမှမလုပ်ခင် Overview Diagram ကို အောက်တွင် ဖော်ပြပေးပါမယ်။



Create & Configure IaC Pipeline using IAC tool terraform

Code Pipeline အတွက် Adminstrator/Cloud Ops မှ terraform ကို အသုံးပြုပြီး AWS Code Pipeline၊ S3 backend with Dynamo DB၊ Github တွေကို setup ပြုလုပ်မှာဖြစ်ပါတယ်။ ဒါတွေကို တည်ဆောက်ဖို့ terraform modules များက အဆင်သင့် ရေးပြီးသားဖြစ်ပါတယ်။ Github က တော့ ကျွန်တော်တို့ ရဲ့ Infra code တွေကို versioning ပြုလုပ်ပြီး ထိန်းသိမ်းဖို့ ဖြစ်ပါတည်။ S3 နဲ့ DynamonDB ကတော့ Terraform backend အတွက်ဖြစ်ပါတယ် အသေးစိပ်ကို **ဒီမှာ** ဖတ်ကြည့်နိုင်ပါတယ်။ Code Pipeline ကတော့ git push လုပ်လိုက်တဲ့ infra code တွေကို Pipeline က ဖြတ်စေပြီး Release or Deploy လုပ်ဖို့ ဖြစ်ပါတယ်။

ဒါကတော့ လိုအပ်တဲ့ Terraform source code တွေရှိတဲ့ github repo ဖြစ်ပါတယ်။ လိုက်စမ်းကြည့်ဖို့ folk/clone ပြုလုပ်နိုင်ပါတယ်။

https://github.com/aws-user-group-myanmar-aws-ugm/infrastructure-as-code-with-aws

```
terraform-demo — Vim main.tf — 80×22
module "s3backend" {
  source        = "DTherHtun/s3backend/aws"
  version       = "0.1.0"
  namespace     = var.backend.namespace
  principal_arn = var.backend.arn
}

module "codepipeline" {
  source    = "DTherHtun/codepipeline/aws"
  name      = "terraform-pipeline"
  vcs_repo = var.vcs_repo
  environment = {
    AWS_ACCESS_KEY_ID     = var.aws.access_key
    AWS_SECRET_ACCESS_KEY = var.aws.secret_key
    CONFIRM_DESTROY       = 0
  }
  s3_backend_config = module.s3backend.config
}
~
~
~
```

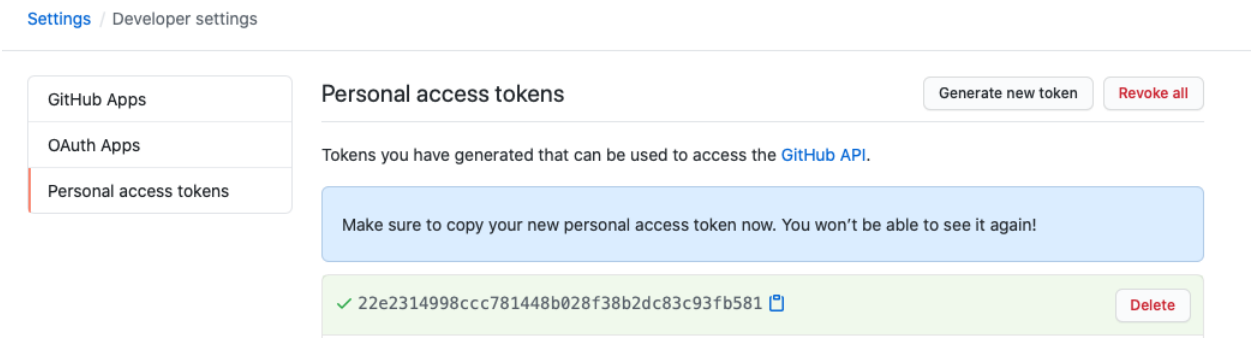
main.tf

အထက် main.tf ပုံမှာ ဖေါ်ပြထားတာကတော့ repo ထဲမှာ terraform-demo folder ထဲတွင် တည်ရှိပါတယ်။ module များကိုတော့ terraform registry မှာ ရရှိနိုင်ပါတယ်။ source = ရဲ့ value မှာ အသုံးပြုမဲ့ terraform modeule ရဲ့ path ဖြစ်ပါတယ်။ module ရဲ့ input / output တွေကိုတော့ terraform registry မှာ တွေ့နိုင်ပါတယ်။

https://registry.terraform.io/modules/DTherHtun/s3backend

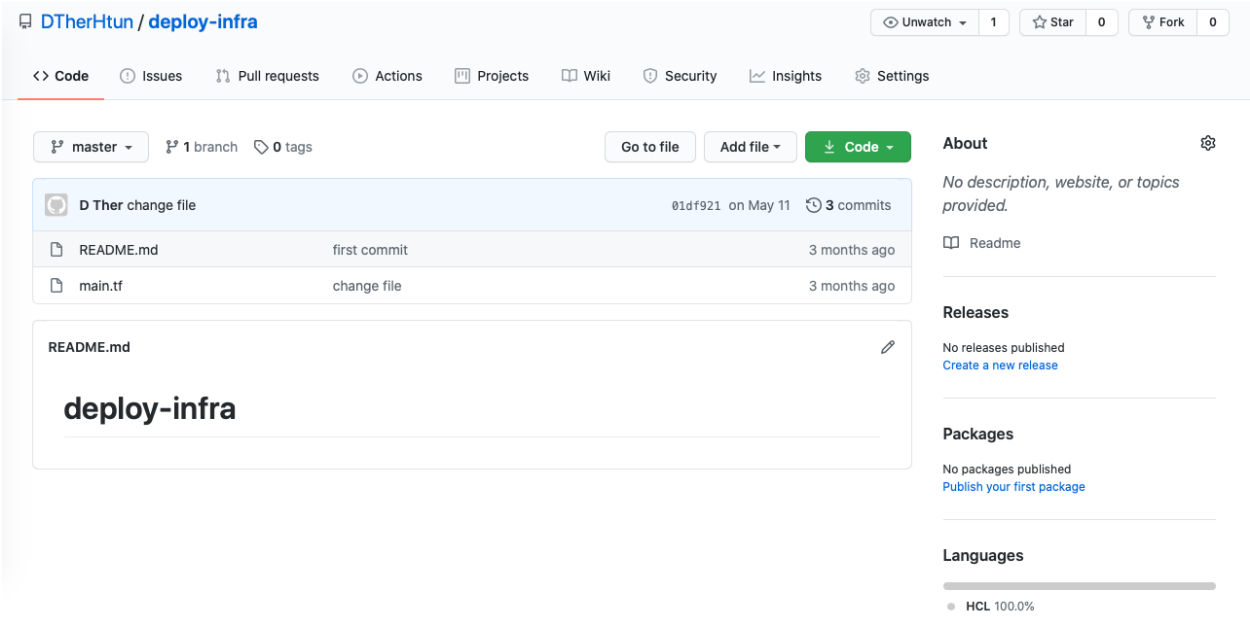
https://registry.terraform.io/modules/DTherHtun/codepipeline

လိုအပ်ချက်တွေ ကိုတော့ variables အနေနဲ့ ထည့်ပေးရမှာဖြစ်ပါတယ်။ terraform.tfvars ကတော့ terraform ရဲ့ variables တွေ ထားတဲ့ file ဖြစ်ပါတယ်။ လိုအပ်ချက်တွေကတော့ aws ရဲ့ resoruces တွေကို တည်ဆောက်နိုင်တဲ့ permission ရှိတဲ့ access key / secret key နဲ့ region လိုအပ်ပါတယ်၊ s3backend အတွက် terraform backend ရဲ့ namespace (workspace) နဲ့ User ရဲ့ arn လိုအပ်ပါတယ်၊ code pipeline အတွက် ချိတ်ဆက်မဲ့ source repo ရဲ့ oauth_token၊ repo name နဲ့ branch တွေလိုအပ်ပါတယ်။



oauth token

oauth_token အတွက် Github ရဲ့ Settings -> Developer settings ထဲက Personal access tokens တွင် အထက်ပုံတွင်ဖေါ်ပြတဲ့ ထားတဲ့ အတိုင်း ပြုလုပ်နိုင်ပါတယ်။



Repo for Infra code

ဒီ repo ကတော့ pipeline နဲ့ ချိတ်ဆက်မဲ့ repo ဖြစ်ပါတယ်။ Engineer များမှ စုပေါင်း အသုံးပြုရမဲ့ repo ဖြစ်ပါတယ်။

```
terraform-demo — Vim terraform.tfvars — 80×22
backend = {
  namespace = "k8smm"
  arn       = "arn:aws:iam::344475516279:user/detherhtun"
}

aws = {
  access_key = "AKIAVANCXHV372UFFJUP"
  secret_key = "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"
  region     = "us-east-2"
}

vcs_repo = {
  branch      = "master"
  identifier  = "DTherHtun/deploy-infra"
  oauth_token = "22e2314998ccc781448b028f38b2dc83c93fb581"
}
~
~
~
~
~
```

terraform.tfvars

ဒီ terraform.tfvars ကတော့ အပေါ်မှာ ပြောခဲ့တဲ့ လိုအပ်ချက်တွေကို ထည့်သွင်းပေးထားတဲ့ variables file ဖြစ်ပါတယ်။ နမူနာပြထားတာ ဖြစ်ပါတယ်။

ဒါဆိုရင် ကျွန်တော်တို့ terraform init ပြုလုပ်တော့မှာ ဖြစ်ပါတယ်။

Terraform init

terraform init ပြုလုပ်ပြီး code တွေအဆင်ပြေမပြေ၊ ဘာတွေ တည်ဆောက်သွားမှာလဲဆိုတာကို မတည်ဆောက်ခင် terraform plan နဲ့ စစ်ကြည့်မှာ ဖြစ်ပါတယ်။

Terraform apply

နောက်ဆုံးမှာတော့ terraform apply command နဲ့ s3backend/codepipeline တို့ကိုတည်ဆောက် configure ပြုလုပ်သွားမှာဖြစ်ပါတယ်။

Code Pipeline Approval

ပထမဆုံး အကြိမ်မှာတော့ ဒီ repo ထဲမှာ ရှိတဲ့ infra code တွေကို တစ်ခါထဲ Pipeline ကနေ တည်ဆောက်သွားမှာ ဖြစ်ပါတယ်။ Pipeline မှာ video ထဲမှာ ပါတဲ့ အတိုင်း တကယ် မတည်ဆောက်ခင် plan ကို review လုပ်ပြီး approve ပြုလုပ်ပေးရမှာ ဖြစ်ပါတယ်။

DTherHtun / deploy-infra

Unwatch 1

Code | Issues | Pull requests | Actions | Projects | Wiki | Security | Insights | Settings

master | deploy-infra / main.tf | Go to file

D Ther change file | Latest commit 01df921 on May 11 | History

0 contributors

20 lines (16 sloc) | 365 Bytes | Raw | Blame

```
provider "aws" {
  version = "2.12.0"
  region  = "us-west-2"
}

data "aws_ami" "ubuntu" {
  most_recent = true

  filter {
    name   = "name"
    values = ["ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-*"]
  }

  owners = ["099720109477"]
}

resource "aws_instance" "helloworld" {
  ami           = data.aws_ami.ubuntu.id
  instance_type = "t2.micro"
}
```

approve ပြုလုပ်ပြီးတဲ့ နောက်မှာ deploy-infra ထဲမှာ ရှိသော terraform infra code အတိုင်း EC2 Instance တစ်လုံးကို us-west-2 region မှာဆောက်ပေးသွားမှာ ဖြစ်ပါတယ်။

aws | Services | Resource Groups | detherhtun @ th3lolipop | Oregon | Support

New EC2 Experience
Tell us what you think

EC2 Dashboard New
Events New
Tags
Limits

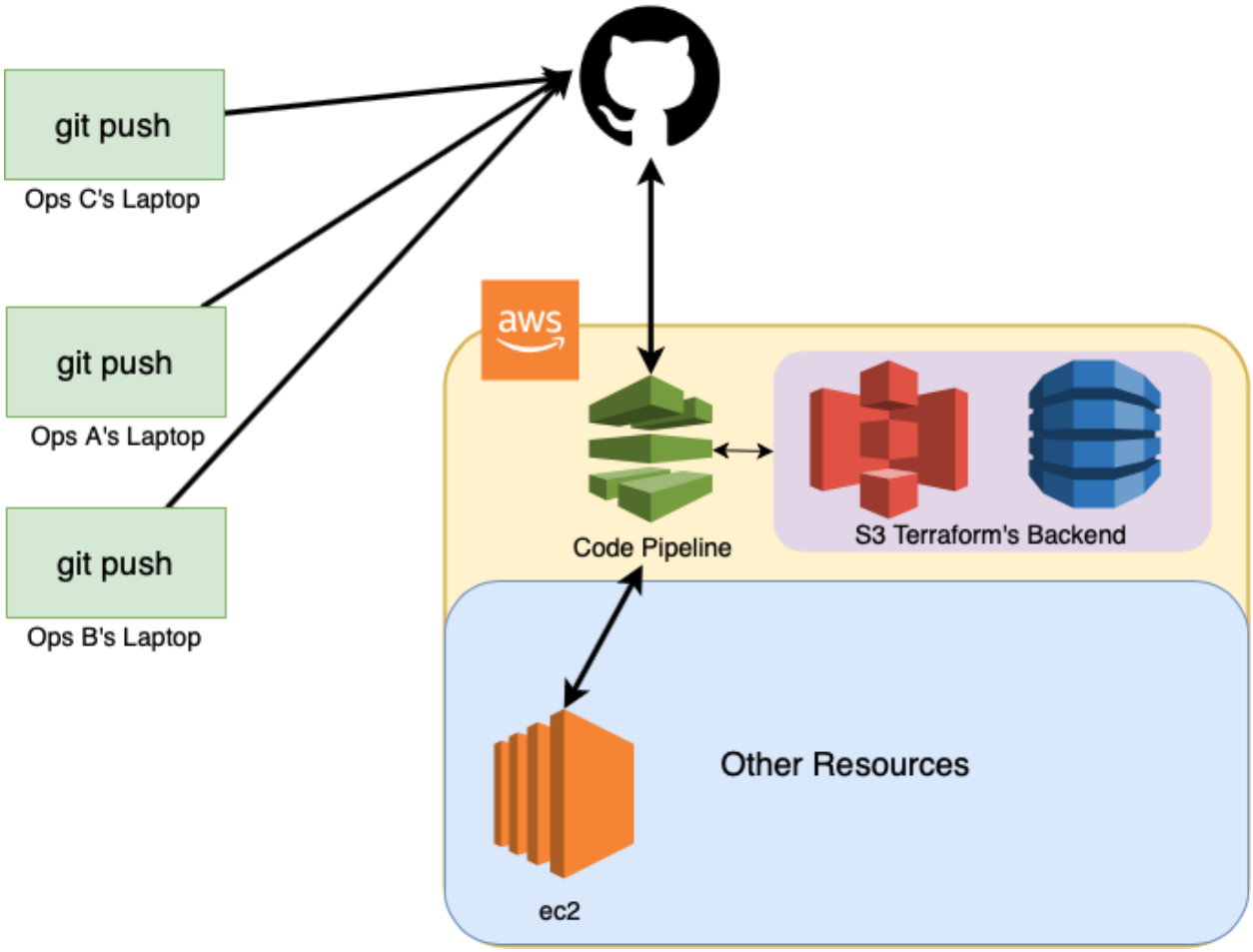
Launch Instance | Connect | Actions

Filter by tags and attributes or search by keyword | 1 to 1 of 1

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks	Alarm Status	Public DNS (IPv4)	IPv4 Public IP	IPv6 IPs	Key Name
	i-0aa3aa4e5e3abd4...	t2.micro	us-west-2c	running	2/2 checks ...	None	ec2-34-219-98-62.us-w...	34.219.98.62	-	

running ec2 via code pipeline

ဒါဆိုရင်တော့ နောက်ပိုင်မှာ Team ထဲမှာ Engineer များ၊ Ops များ က deploy-infra git repo ကို အသုံးပြုပြီး Code Pipeline မှ တဆင့် Infrastructure ကို develope ဆက်လုပ်နိုင်မှာ ဖြစ်ပါတယ်။



စာလည်းတော်တော်ရှည်သွားပြီဆိုတော့ နိဂုံးချုပ်ရမယ်ဆိုရင် မတူညီတဲ့ IaC နှစ်မျိုး နဲ့ AWS CodePipeline ဆောက်ပြီး AWS Resources တွေ Automated Build ဖို. ကြိုက်တဲ့ ပုံစံမျိုးကိုသုံးပြီး တည်ဆောက်လို.ရတာလေးကို ဒီ tutorial လေးမှာ
ဖော်ပြပေးသွားတာဖြစ်ပါတယ်။တစ်ခုခုလိုအပ်လို.ပဲဖြစ်ဖြစ် အမှားအယွင်းတစ်ခုခုပါသွားခဲ့ရင်ဖြစ်ဖြစ် ကျွန်တော်တို.ကို အကြံပြုတင်ပြနိုင်ပါတယ် ဟုပြောရင်း ဒီ tutorial လေးကို နိဂုံးချုပ်ပါရစေ။

အောက်က URL တွေမှာ Reference ယူပြီးကိုတိုင်လိုအပ်သလို ပြင်ဆင်နိုင်ပါတယ်။
အားလုံးကိုကျေးဇူးတင်ပါသည်။

https://github.com/aws-user-group-myanmar-aws-ugm/infrastructure-as-code-with-aws

References:
https://aws.amazon.com/about-aws/whats-new/2016/11/aws-codepipeline-introduces-aws-cloudformation-deployment-action/
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
https://docs.aws.amazon.com/codepipeline/latest/userguide/welcome.html

Also Dedicated Thanks to **Ko D Ther Htun (dtherhtun.cw@gmail.com)**

PhyoMinHtun @ AWS User Group Myanmar phyominhtun1990@gmail.com

IoT Rules Engine

AWS IoT Core

ဒီနေ့ ခေတ်မှာ Smart City တွေ ဖြစ်လာတာနဲ့အမျှ Internet နဲ့ချိတ်ဆက်ထားတဲ့ Devices တွေ Things တွေကိုလည်း နေရာတိုင်းမှာတွေ့လာရပါတယ်။

ဒီနေ့မှာတော့ အဲ့လို Internetနဲ့ချိတ်ထားတဲ့ things တွေကို AWS ရဲ့ Internet of thingsအပိုင်းမှာပါတဲ့

AWS IoT Core သုံးပီး ဘယ်လို Control လုပ်လို့ရနိုင်လဲ ဘယ်လို Actions တွေထည့်လို့ ရလဲဆိုတာ ကိုပြောပြပါရစေ။

AWS IoT ဘာကြောင့်သုံးလဲဆိုရင်

IoT Core က ရလာတဲ့ Raw Datas တွေကို SQL Rule query statement တွေနဲ့ပြန်ပီး filter လုပ်လို့ရနိုင်ပါတယ်။

Real time တစ်ခါတည်း IoT Topic နောက်တစ်ခုကို တန်းပီး Republish လုပ်လို့ရနိုင်ပါတယ်။

Raw data တွေ ဖြစ်တဲ့ Messages တွေကို AWS service သုံးပီး Data တွေ S3, DynamoDB ,etc တွေမှာ store လုပ်နိုင်ပါတယ်။

Store လုပ်ထားတဲ့ Data တွေကို Kinesis Analytics တွေ QuickSight တို့လို Analytic tool တွေ တစ်ခါတည်း သုံးပီး Real time monitoring Dashboard ပါတစ်ခါတည်းလုပ်နိုင်တာကလည်း AWS IoT core ရဲ့ အားသာချက်တွေ ဖြစ်ပါတယ်။

SQL (Rule Query Statement) နဲ့ Filter လုပ်ထားတဲ့ Message တွေကို ကိုယ်ရဲ့ Application မှာ တကယ်လိုအပ်တဲ့ပုံစံကို AWS Service တွေဖြစ်တဲ့

- S3
- DynamoDB
- Lambda Function
- SNS
- SQS
- Kinesis Stream
- Kinesis Firehose Stream
- IoT Topic
- CloudWatch
- Elastic Search
- Salesforce IoT Input Stream

- IoT Events input
- Step Function
- HTTPS

Services တွေနဲ့ **real time** ချိတ်ဆက်ပီး အသုံးပြုလို့ရပါတယ်။

IoT (Internet of Things)

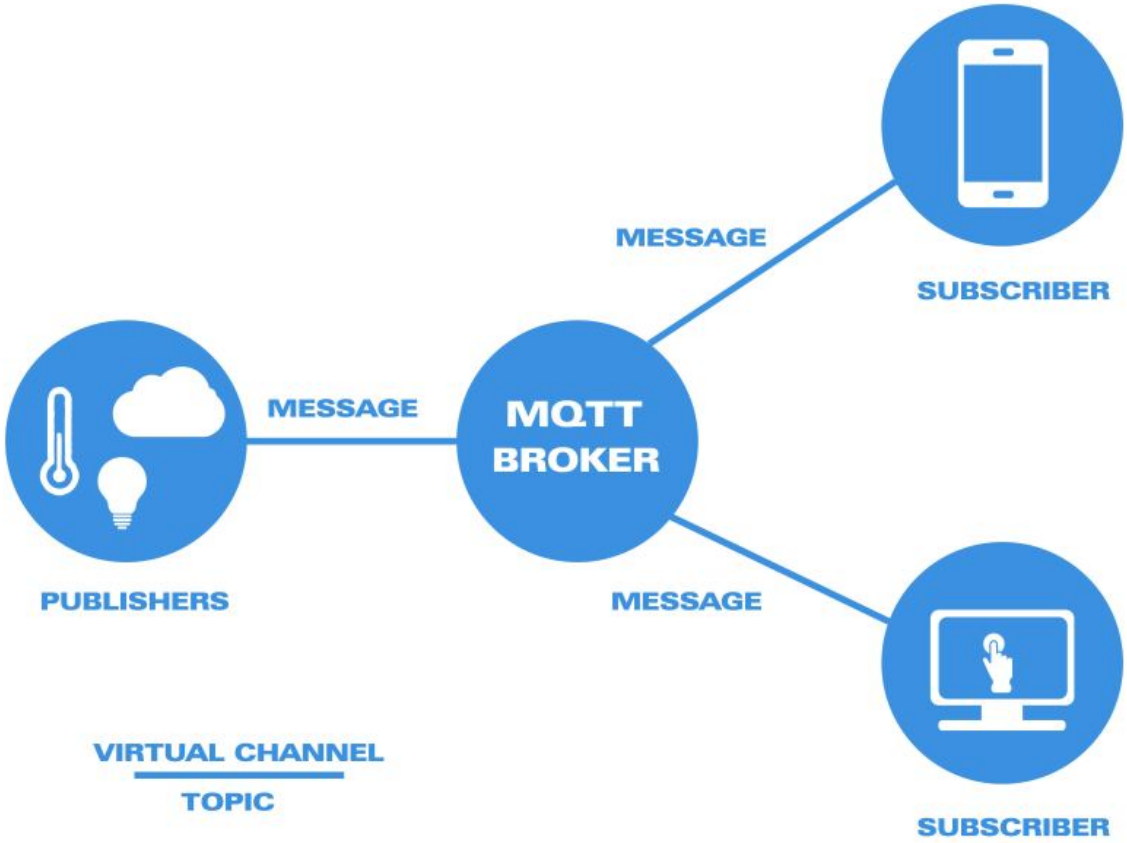
IoT Devices

ပထမဦးစွာ IoT Devices တွေကို AWS IoT Core ရဲ့ Manage မှာ Things connect လုပ်ပီး အသုံးပြုလို့ရနိုင်ပါတယ်။

ဒီ Article မှာတော့ Device မပါပဲ *MQTT Client* ကို အသုံးပြုပီး *Topic* ကို *Subscribe / Publish* လုပ်သွားပါမယ်။

MQTT (Message Queuing Telemetry Transport)

MQTT ဆိုတဲ့ Message Queuing Telemetry Transport ဆိုတဲ့ Protocol ကို တော့ IoT သုံးဖူးတဲ့သူတိုင်း သိကြမှာပါ။



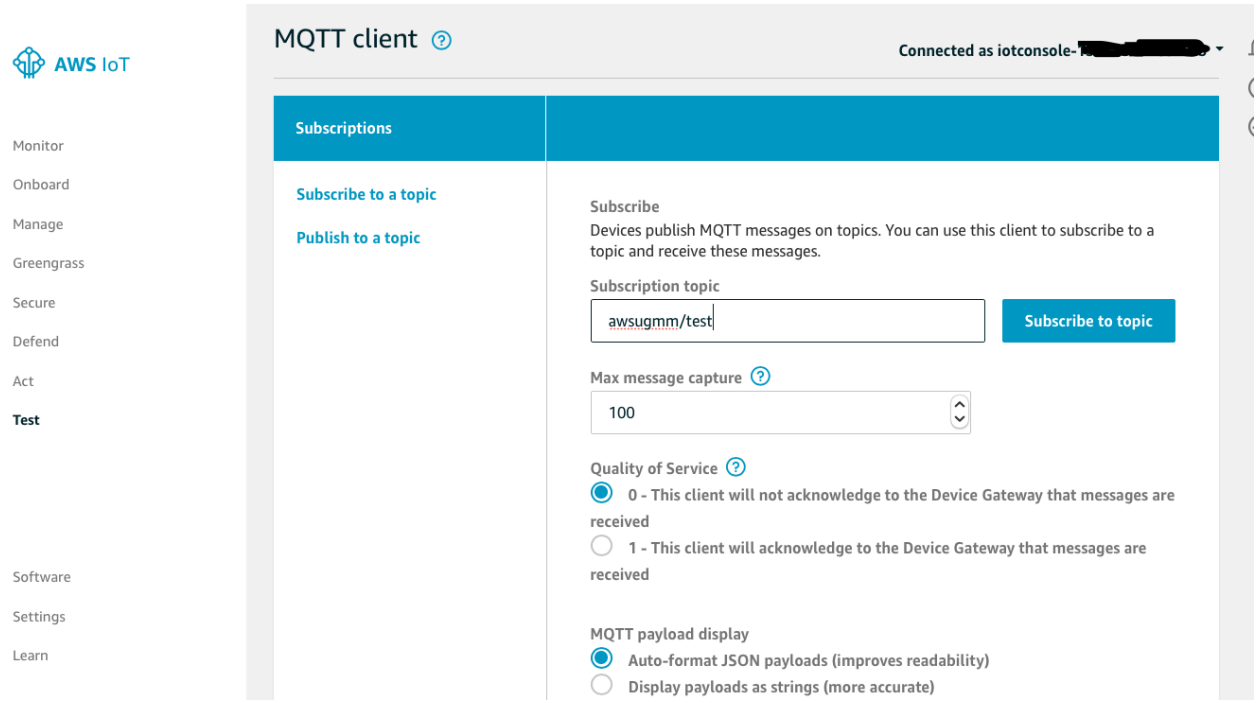
MQTT

MQTT ရဲ့ အလုပ်လုပ်ပုံကတော့ Publisher ဆိုတာ IoT devices တွေကနေ Real time messages တွေကို ပို့လိုက်ပီး Subscriber က လက်ခံသူတွေ ဖြစ်ပါတယ်။

အဲ့ဒီ MQTT ကရလာတဲ့ raw message ကို AWS IoT Coreသုံးပီး Rules engine မှာ SQL နဲ့ ဘယ်လို query လုပ်လို့ရနိုင်လဲဆိုတာ နားလည်သလောက်ပြန်ရှင်းပြပါရစေ။

MQTT Client (Test)

Subscribe

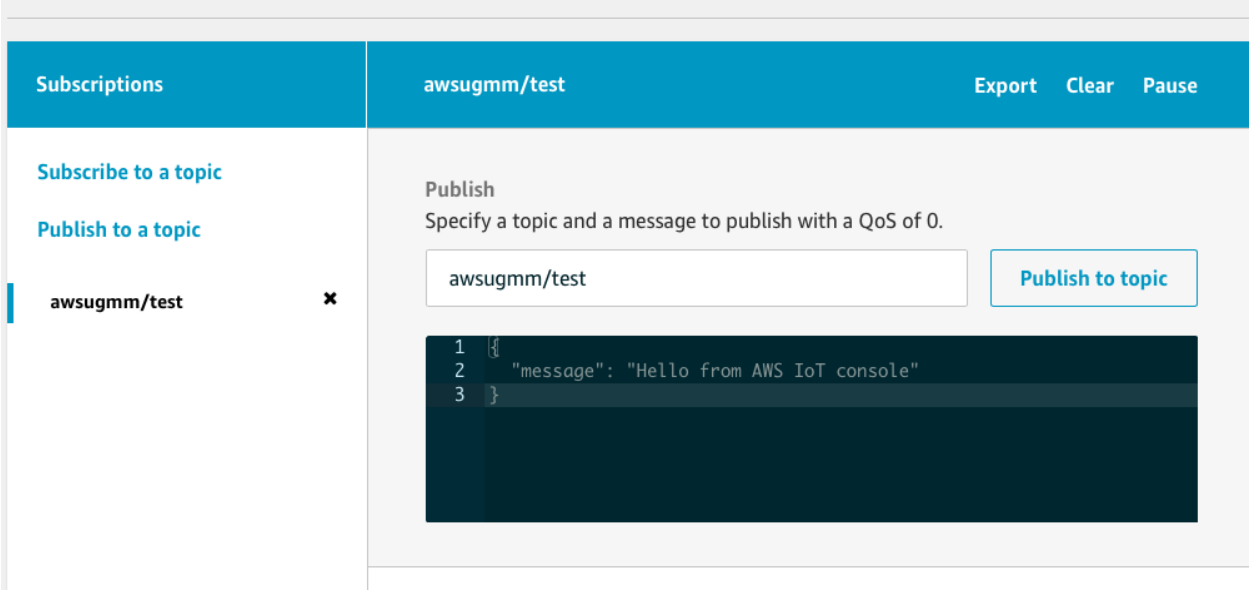


ပုံမှာပြထားသလို AWS IoT Core အောက်မှာ Testလုပ်လို့ရနိုင်ပါတယ်။

အခု စမ်းပြထားတာကတော့ **MQTT client မှာ awsugmm/test** ဆိုတဲ့ topic ကို subscribe လုပ်ထားတာပါ။

Subscribe လုပ်ထားပီးရင် အောက်ကပုံလို မြင်ရမှာပါ။

Publish



အဲ့ဒီ JSON Formatနဲ့ပြထားတဲ့နေရာကနေ message တွေကို publish လုပ်ပြပါမယ်။ တကယ့် လက်တွေ့ IoT Device နဲ့စမ်းရင်တော့ အဲ့ IoT Device ကရလာတဲ့ Message ကို ဒီ MQTT Client ကနေ real time data တွေကို မြင်ရမှာပါ။

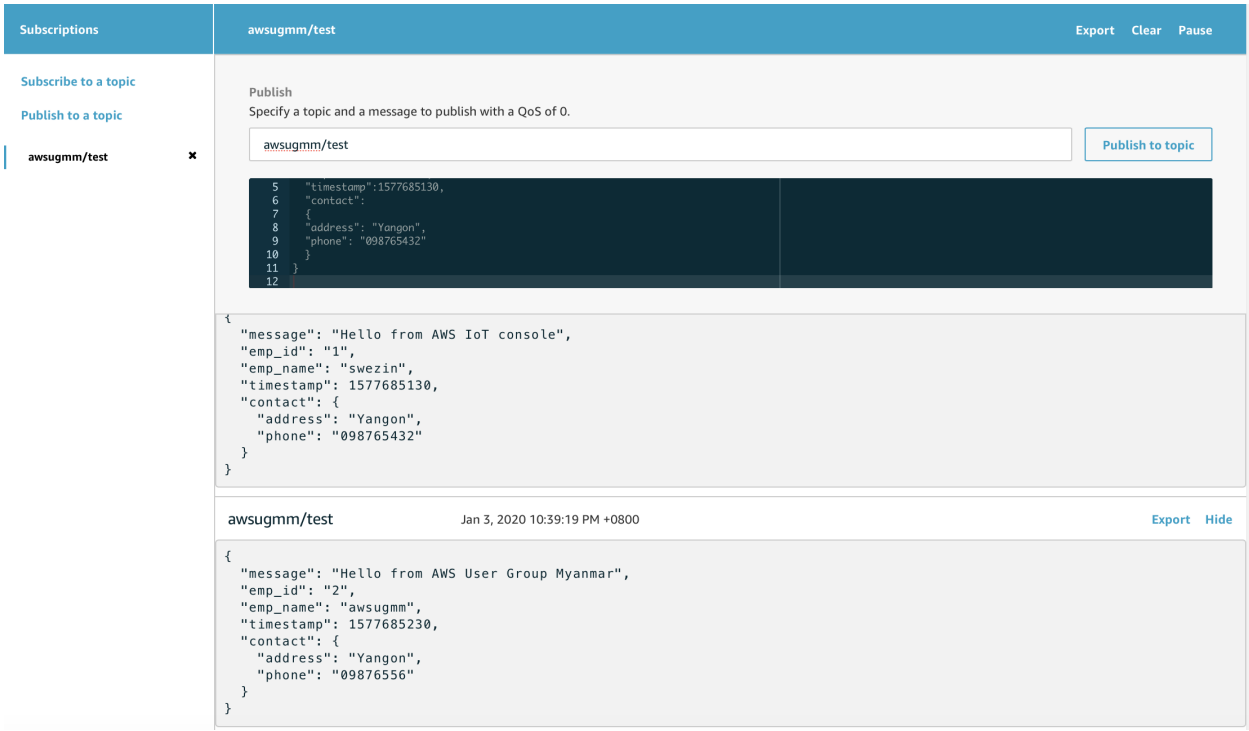
အခုကတော့

Demo


```
{
 "message": "Hello from AWS IoT console",
 "emp_id": "1",
 "emp_name": "swezin",
 "timestamp":1577685130,
 "contact":
 {
 "address": "Yangon",
 "phone": "098765432"
 }
}

{
 "message": "Hello from AWS User Group Myanmar",
 "emp_id": "2",
 "emp_name": "awsugmm",
 "timestamp":1577685230,
 "contact":
```

```
 {
 "address": "Yangon",
 "phone": "09876556"
 }
}
```

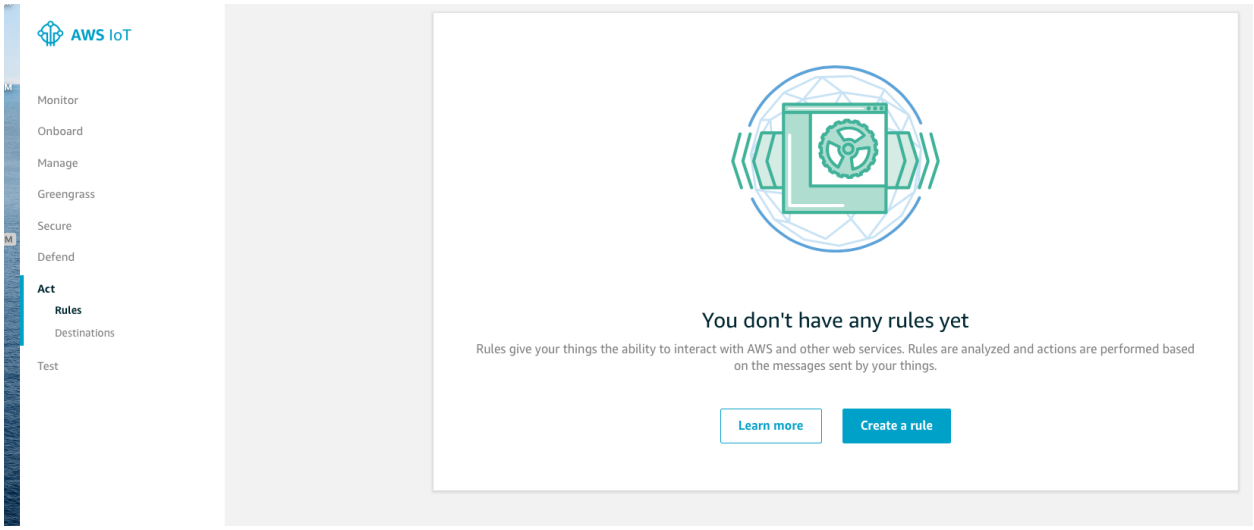
JSON Format နဲ့ Data ကို အောက်ထဲကပုံကလို Publish လုပ်ပြထားပါတယ်။



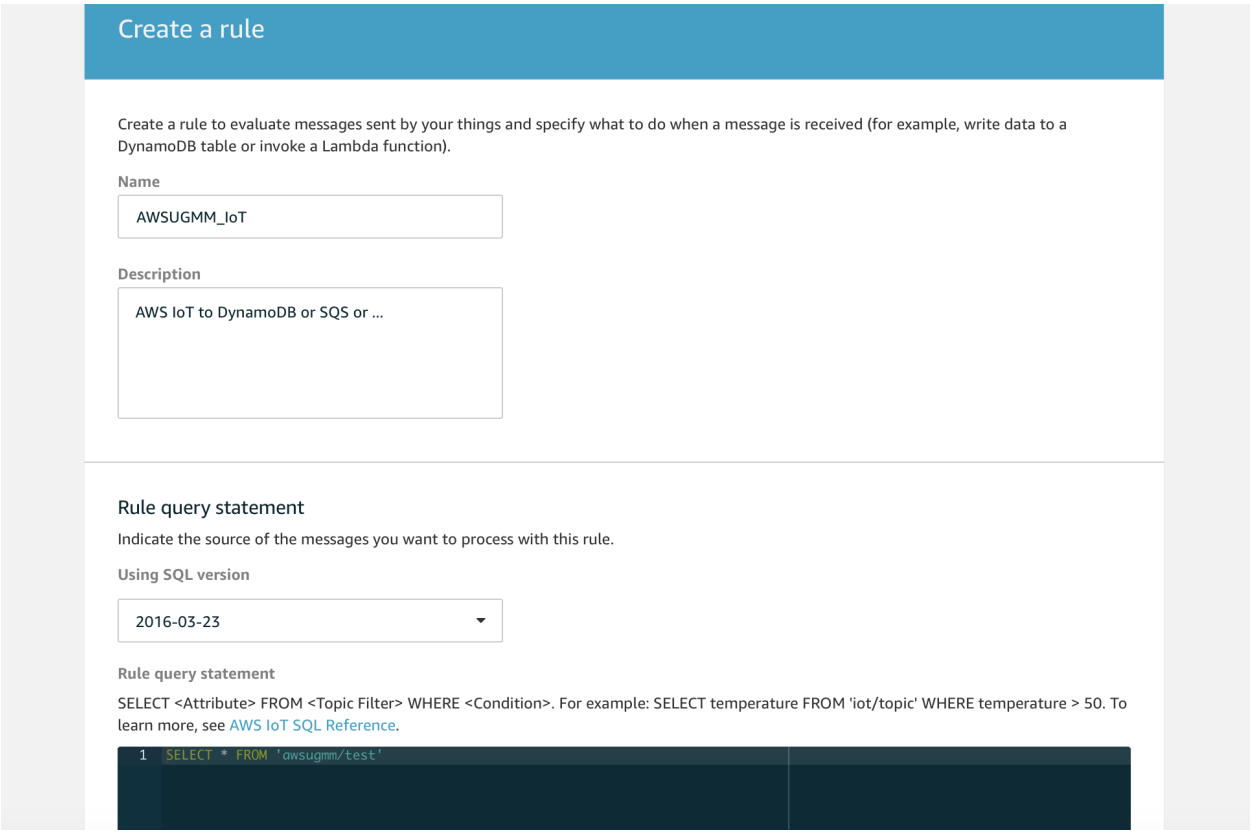
IoT Rules

Rules Engine / Rule actions လို့လည်း ခေါ်လို့ရပါတယ်။

ဘာ Rule မမှ ရှိသေးတဲ့ပုံပါ။



Rule တစ်ခု Create ဖို့ဆိုရင် Topic တစ်ခုလောက်ရှိထားသင့်ပါတယ်။ Test ကနေ topic ကို publish/subscribe လုပ်ထားပီး စမ်းကြည့်လို့ရနိုင်ပါတယ်။



Name တွေ Description တွေက မိမိ ကြိုက်တာပေးလို့ရပါတယ်။ ဒီ Article ရဲ့ အဓိက ပြောပြချင်တဲ့အပိုင်းလေးကို ရောက်ပါပြီ။ တစ်ခုကတော့ **Rule query statement** ပါ။

Rule Query Statement

SELECT * FROM 'awsugmm/test'

က တော့ Topic ကရလာတဲ့ Data အကုန်လုံးကို ယူထားတာပါ။

IoT Rule engine က JSON Data types အကုန်လုံးကို support လုပ်ပေးသလို Int/ Decimal ကနေ String , String ကနေ Decimal အကုန်ပြောင်းလို့ရပါတယ်။ နောက်အားသာချက်တွေက Functions တွေ Operators တွေ Literals တွေ Case statements တွေပါ supportပေးတော့ သုံးရတာ တော်တော် အဆင်ပြေပါတယ်။

အသေးစိတ်ကိုတော့ ဒီ link (https://docs.aws.amazon.com/iot/latest/developerguide/iot-sql-reference.html)ကနေ လေ့လာနိုင်ပါတယ်။

SELECT Clause
FROM Clause
WHERE Clause

အပေါ်က SQL Clause 3ခုကိုတော့ သိထားသင့်ပါတယ်။ Rule createလုပ်ပီးရင် တော့နည်းနည်း စမ်းပြပါမယ်။

Rule Actions

အပေါ်က ပြောခဲ့တဲ့ AWS Serviceတွေကို action ကနေ သွားထည့်လို့ရပါတယ်။ Rule create လုပ်တဲ့အချိန်မှာ Add actions ကိုနှိပ်လိုက်ရင် အောက်က ပုံကို တွေ့ရမှာပါ။

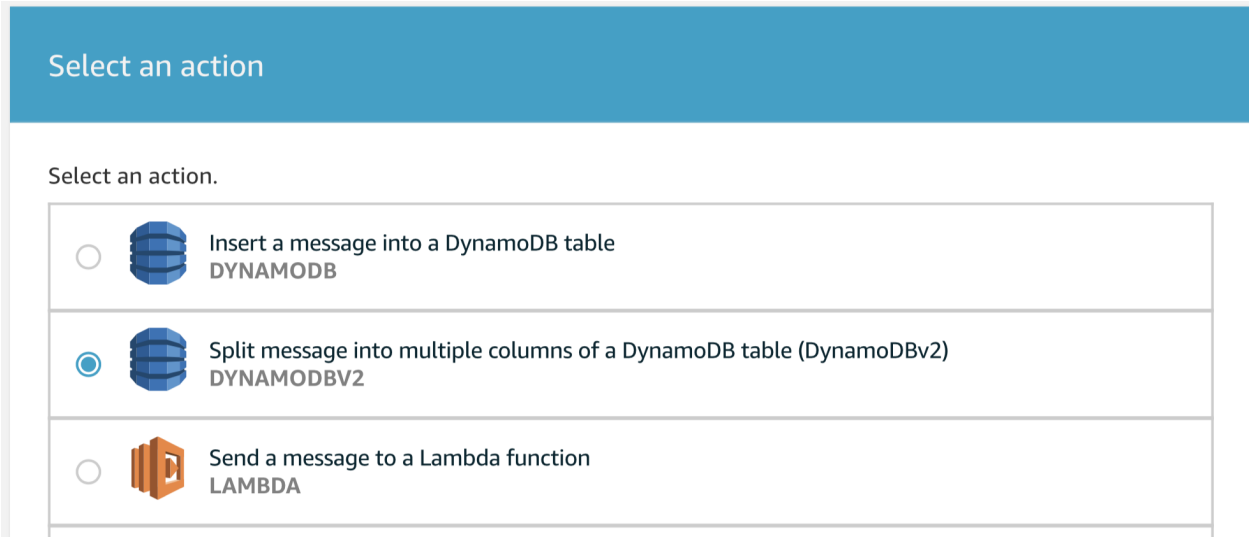
Select an action

Select an action.

- ○ Insert a message into a DynamoDB table
  DYNAMODB
- ○ Split message into multiple columns of a DynamoDB table (DynamoDBv2)
  DYNAMODBV2
- ○ Send a message to a Lambda function
  LAMBDA
- ○ Send a message as an SNS push notification
  SNS
- ◉ Send a message to an SQS queue
  SQS
- ○ Send a message to an Amazon Kinesis Stream
  AMAZON KINESIS
- ○ Republish a message to an AWS IoT topic
  AWS IOT REPUBLISH
- ○ Store a message in an Amazon S3 bucket
  S3
- ○ Send a message to an Amazon Kinesis Firehose stream
  AMAZON KINESIS FIREHOSE
- ○ Send message data to CloudWatch
  CLOUDWATCH METRICS
- ○ Change the state of a CloudWatch alarm
  CLOUDWATCH ALARMS
- ○ Send a message to the Amazon Elasticsearch Service
  AMAZON ELASTICSEARCH
- ○ Send a message to a Salesforce IoT Input Stream
  SALESFORCE IOT
- ○ Send a message to IoT Analytics
  IOT ANALYTICS

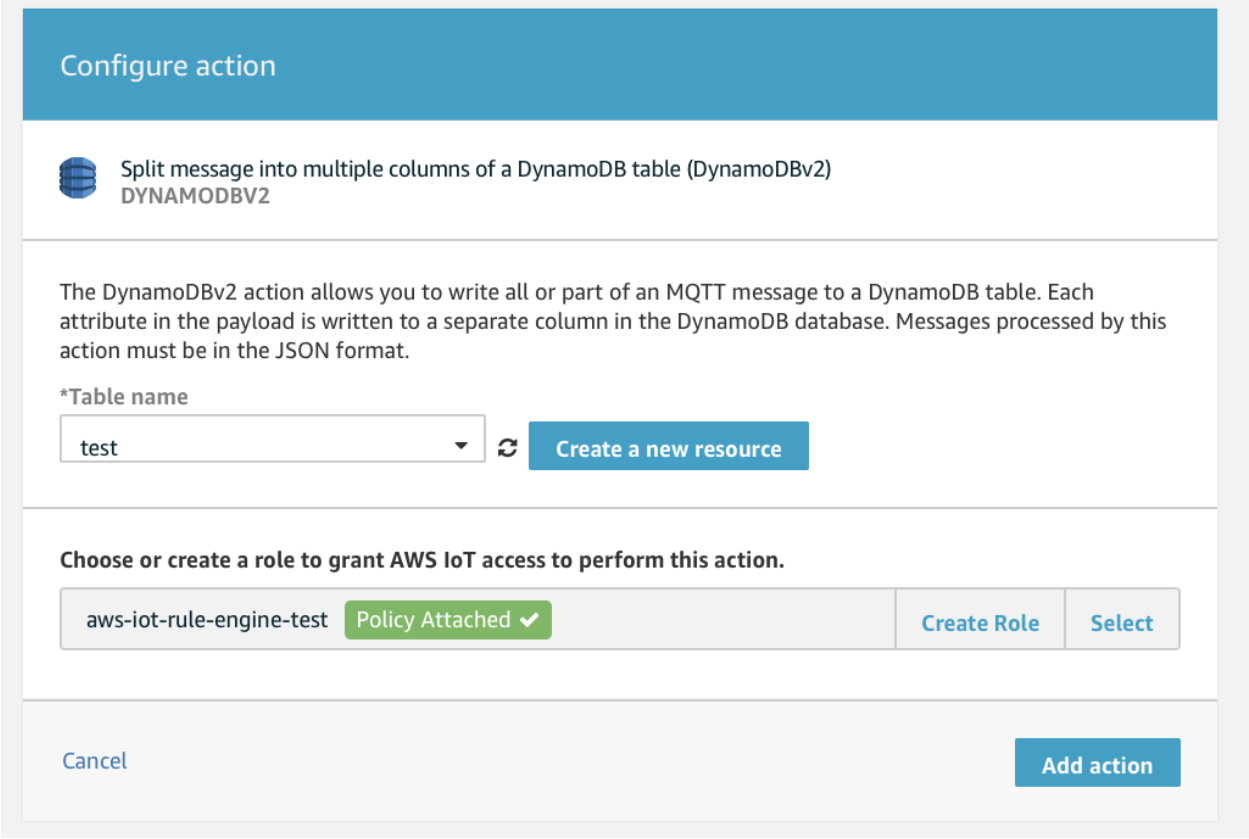
ဒီလို Actions တွေ ကို သုံးပီး မိမိတို့ရဲ့ IoT Devices ရဲ့ Data တွေကို Real time monitoring / Dashboard ပါတစ်ခါတည်းလုပ်နိုင်ပါတယ်။

အခုကတော့ DynamoDBနဲ့စမ်းပြပါမယ်။



Choose an action

Action တစ်ခုကို ရွေးပီးရင် မိမိသုံးမယ့် Action က ဒီ AWS IoT Core ကနေ လာတဲ့ Datas တွေကိုလက်ခံမယ့် Resourceရှိဖို့လိုပါတယ်။ မရှိသေးလို့ Createလုပ်လို့ရပါတယ်။ ဒီ Article မှတော့ test ဆိုတဲ့ DynamoDB tableနဲ့ နည်းနည်း စမ်းပြပါမယ်။



Action တစ်ခုကိုထည့်ပီးရင် အောက်ကပုံလို နောက် Action တွေပါထပ်ထည့်လို့ရပါတယ်။

Cloud Object Storage (aka) Amazon Simple storage service (S3)

Cloud Object Storage

မနက်မိုးလင်းတော့ Cloud Front က Object Cache လေးကို invalidate လုပ်နေတုန်း စဉ်းစားကြည့်တော့ Object Storage အကြောင်းလေးတွေးမိရာကနေ AWS S3 အကြောင်းရေးမယ်ပေါ့ ပီတော့ ကိုယ့်လူ ဝေယံမင်း ကလည်း ကိုဖြိုး S3 အကြောင်းလေးရေးရအောင် စာမေးပွဲအတွက်လည်း ပြင်ဆင်နေတုန်းလေးဆိုတော့ရေးမယ်ပေါ့ ဘာညာနဲ့ ဒီ article လေးကိုရေးမိသွားတယ်။

AWS S3 ကို မပြောခင် Storage Type ဘယ်နှစ်မျိုးရှိလဲ တစ်ချက်လောက်လေ့လာကြည့်ကြည့် ရအောင်...

Storage Type ဘယ်နှစ်မျိုးရှိသလဲ ဆို တော့

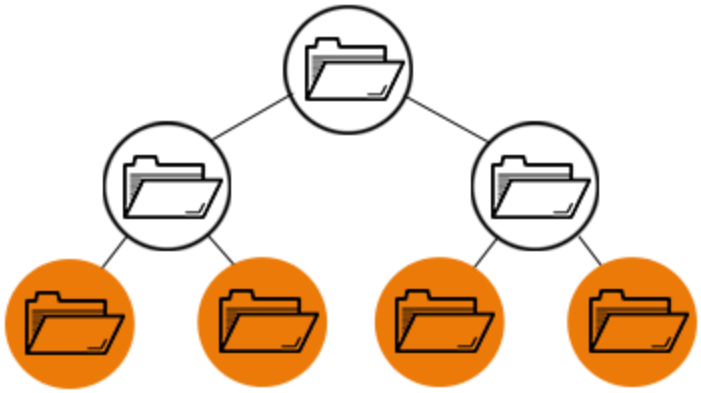
1. File Storage

2. Block Storage

3. Object Storage

ဆိုပြီးတော့ ရှိကြပါတယ်။ အသေးစိတ်ကို အောက်မှာဖော်ပြထားပါတယ်။

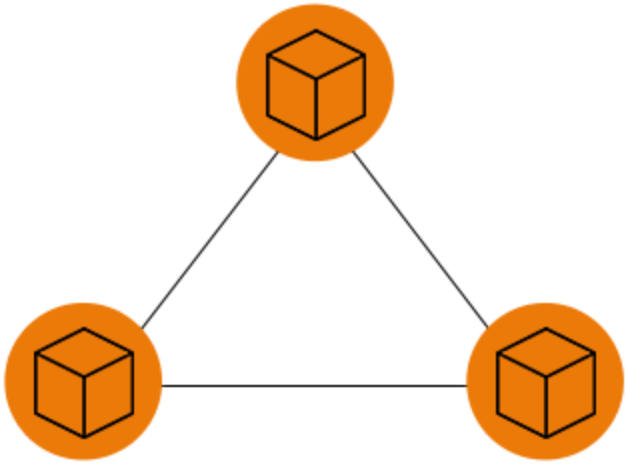
File Storage



File Storage [Image Credit : redhat.com]

File Storage လို ပြောလိုက်တာနဲ့ ကျွန်တော်တို.တော်တော်များများ ရင်းနှီးပီးသားဖြစ်ပါလိမ့်မယ်။ Files တွေ Folders တွေကို Network ကနေတဆင့် Share ပြီးသုံးကြတာပါ။ Data တွေက File Server ရဲ့ folder တစ်ခုမှာ စုဝေးထားပြီးတော့ တခြားသော ကွန်ပြူတာကနေ တဆင့် Data တွေကို သိမ်းဆည်းထားပါတယ်။

Block Storage



Block Storage [Image Credit : redhat.com]

Name

AWSUGMM_IOT

Description

AWS IoT to DynamoDB or SQS or ....

Rule query statement

Indicate the source of the messages you want to process with this rule.

Using SQL version

2016-03-23

Rule query statement

SELECT <Attribute> FROM <Topic Filter> WHERE <Condition>. For example: SELECT temperature FROM 'iot/topic' WHERE temperature > 50. To learn more, see AWS IoT SQL Reference.

```
1 SELECT * FROM 'awsugmm/test'
```

Set one or more actions

Select one or more actions to happen when the above rule is matched by an inbound message. Actions define additional activities that occur when messages arrive, like storing them in a database, invoking cloud functions, or sending notifications. (*.required)

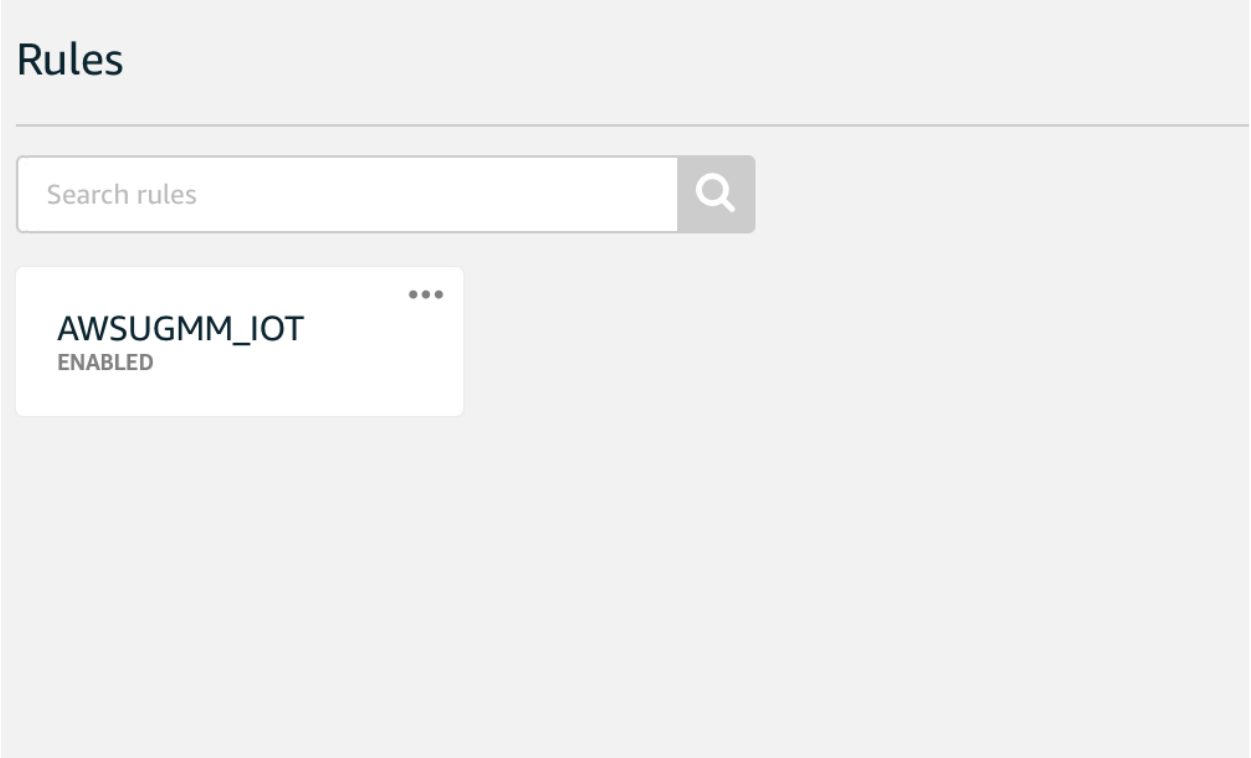
Split message into multiple columns of a DynamoDB table...
test

Remove Edit

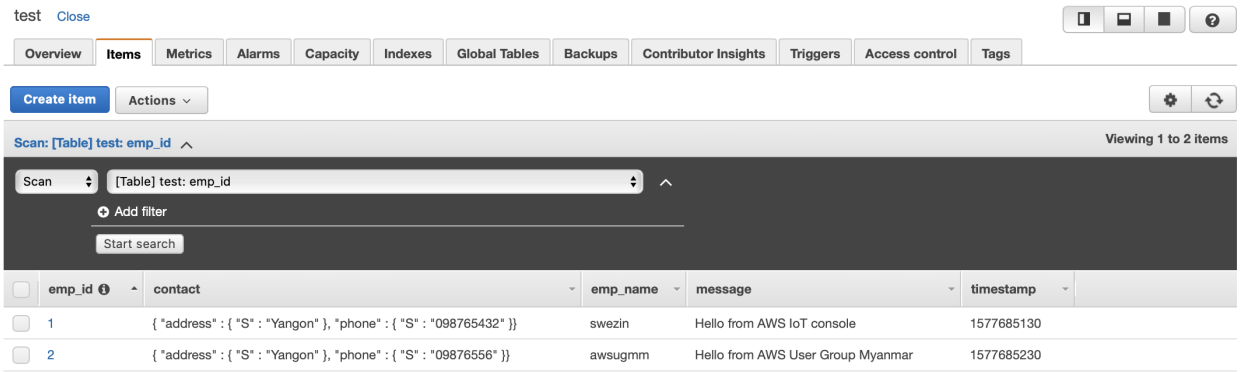
Add action

Rule တစ်ခုမှာ Actions တွေကို ကိုယ့် Solutions မှာလိုအပ်သလို ကြည့်ပီးထည့်လို့ရတာက IoT Core ရဲ့ အားသာချက် နောက်တစ်ခုပါ။

Rule တစ်ခုမှာ error action ပါထည့်လို့ရပါတယ်။ Rule တစ်ခုကို Create လုပ်ပြရင် အောက်ကပုံလို မြင်ရမှာပါ။



Rule Create လုပ်ပီးရင် Test ကနေ Message တွေကို Publishလုပ်ပီး DynamoDB မှာဝင် Action အလုပ်လုပ် /မလုပ်ကို ဝင်ကြည့်လို့ရပါပီ။

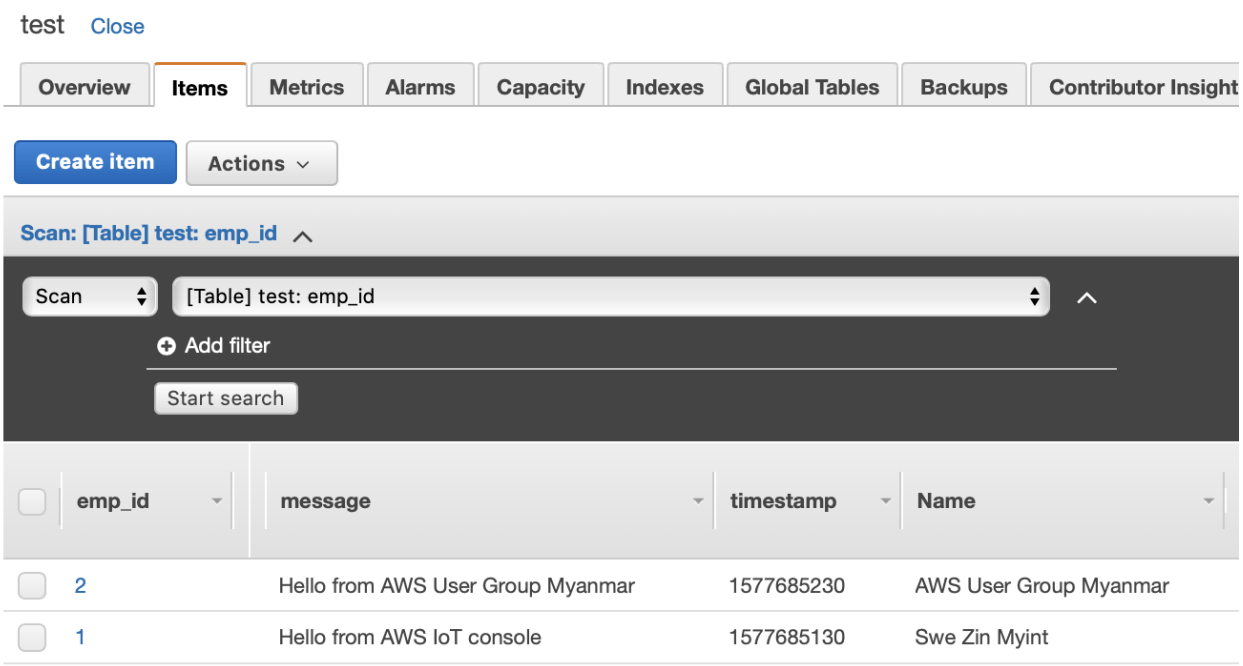


Action လည်း အလုပ်လုပ်ပီဆိုတော့ SQL ရဲ့Case Statmentလေး စမ်းပြပါမယ်။

```
SELECT *,
CASE emp_name
```

WHEN 'swezin' THEN 'Swe Zin Myint'
WHEN 'awsugmm' THEN 'AWS User Group Myanmar'
ELSE 'no name' END as Name
FROM 'awsugmm/test'

Demo စမ်းပြထားတဲ့ emp_name ရဲ့ swezinတို့ awsugmmတို့ကို Full Nameပြောင်းပီး နောက် Column တစ်ခုကို Case statement နဲ့ create လုပ်ပီး Update လုပ်ပေးလိုက်ပါ။



Test Table မှာ Column တစ်ခု create လုပ်သွားပါတယ်။ ဒီနေရာမှာ ပြောချင်တာကတော့ IoT Devices ကပို့လိုက်တဲ့ Data တွေကို SQL Rule Query statement ကနေ တော်တော်များများကို filter လုပ်နိုင်ပါတယ်။

SELECT *,
topic(2) as level FROM 'awsugmm/+/test'

+ ကတော့ any one level ပါ။ awsugmmအောက်မှာ Devices ပေါင်းစုံရှိနေရင် အဲ့ Device အကုန်လုံးရဲ့ data တွေပါ ရနိုင်ပါတယ်။ topic(2) ကတော့ DynamoDB ထဲကို levelဆိုတဲ့ column အနေနဲ့ ဘယ် Deviceကလာတာကို သိမ်းထားနိုင်ပါတယ်။လေ့လာနေတဲ့သူတစ်ယောက်မို့ အမှားပါရင် ခွင့်လွှတ်ပေးကြပါ။ ဆက်လက်ကြိုးစားပါ့မယ်။

Swe Zin Myint

AWS User Group Myanmar