AWS Cloud ပေါ်က resources တွေကို reliability ကောင်းအောင် ဘယ်လိုလုပ်ကြမလဲ

Reliability Pillar

Reliability ကောင်းအောင် ဘယ်လိုလုပ်ရမလဲမပြောခင် Cloud architecture မှာ Reliability pillar က ဘာကိုဆိုလိုတာလဲ အရင် definition ဖွင့်ကြရအောင် ။

Reliability ဆိုတာက ဒဲ့ပြန်မယ်ဆို ယုံကြည်ကိုးစားနိုင်မှုပေါ့။ ဘယ်အခြေအနေမဆို စိတ်ချပြီး လွှတ်ထားလို့ရတယ်လို့ အလွယ်မြင်တာပေါ့ ။

Cloud ပေါ်မှာတော့ reliability ဆိုတာ system infrastructureပေါ်က server တွေ downသွားရင် မ fail သွားပဲ ပြန်အလုပ်လုပ်နိုင်တာ၊ recover နိုင်တာကိုဆိုလိုပါတယ်။ traffic spike သို့မဟုတ် မြင့်တက်လာတဲ့ workload demand ပေါ်မူတည်ပြီး resources တွေကို အတိုးအလျှော့၊ auto scale in scale out လုပ်နိုင်တာလည်း reliability ကောင်းတာလို့ ပြောနိုင်ပါတယ် ။

ကဲ ဒါဆို Reliability ကောင်းအောင် ဘာလိုလုပ်မလဲ။ ပြောမယ်ဆို services အသေးစိတ်ကျပြီး တာသွားမှာဆိုတော့ whole pillar cover ဖြစ်တဲ့ အပိုင်းတွေ ပြောကြတာပေါ့ ။

Traffic spike ဖြစ်တာတွေ၊ increasing workload demand ကို handle နိုင်အောင် အရင်လုပ်ကြတာပေါ့။ 1/5

Web applications တွေအတွက်ဆို Elastic load balancing ELB, auto scaling တွေသုံးသင့်တယ်။

ELB ဆိုတာက web app တစ်ခု ကို Ec2 instance တစ်လုံးတည်း မှာ တင်ထားတယ်။ အကယ်၍ user request သိန်းချီဝင်လာပြီး သုံးမယ်ဆို traffic spike ဖြစ်ပြီး server fail သွားနိုင်တယ်။ ELB သုံးမယ်ဆို server အနည်းဆုံး ၂လုံးတော့ရှိရမှာပေါ့။ အဲ့တာမှ load balancing လုပ်လို့ရပြီး load balancer ဖြစ်မှာပေါ့ ။ server တစ်လုံး down သွားရင် နောက် တစ်လုံး redundant အနေနဲ့ ရှိအောင်လဲ သုံးကြတယ်။

Auto scaling ဆိုတာက မြင့်လာတဲ့ workload ပေါ်မူတည်ပြီး compute resources တွေ အတိုးအလျော့ scale in scale out လုပ်ပေးတာ။ user requests က EC2 server 2 လုံးနဲ့ မနိုင်တော့ဘူး။ 4 လုံးနဲ့ ကစ်ကွာဆိုပြီး လုပ်ပေးတာပေါ့။ minimum, desired, maximum capacity သတ်မှတ်လို့ရတယ် auto scaling လုပ်မယ်ဆို။ မဟုတ်ရင် တိုးချင်သလိုတိုးဆို ကုန်ချင်သလောက်ကုန်မှာပေါ့ :3

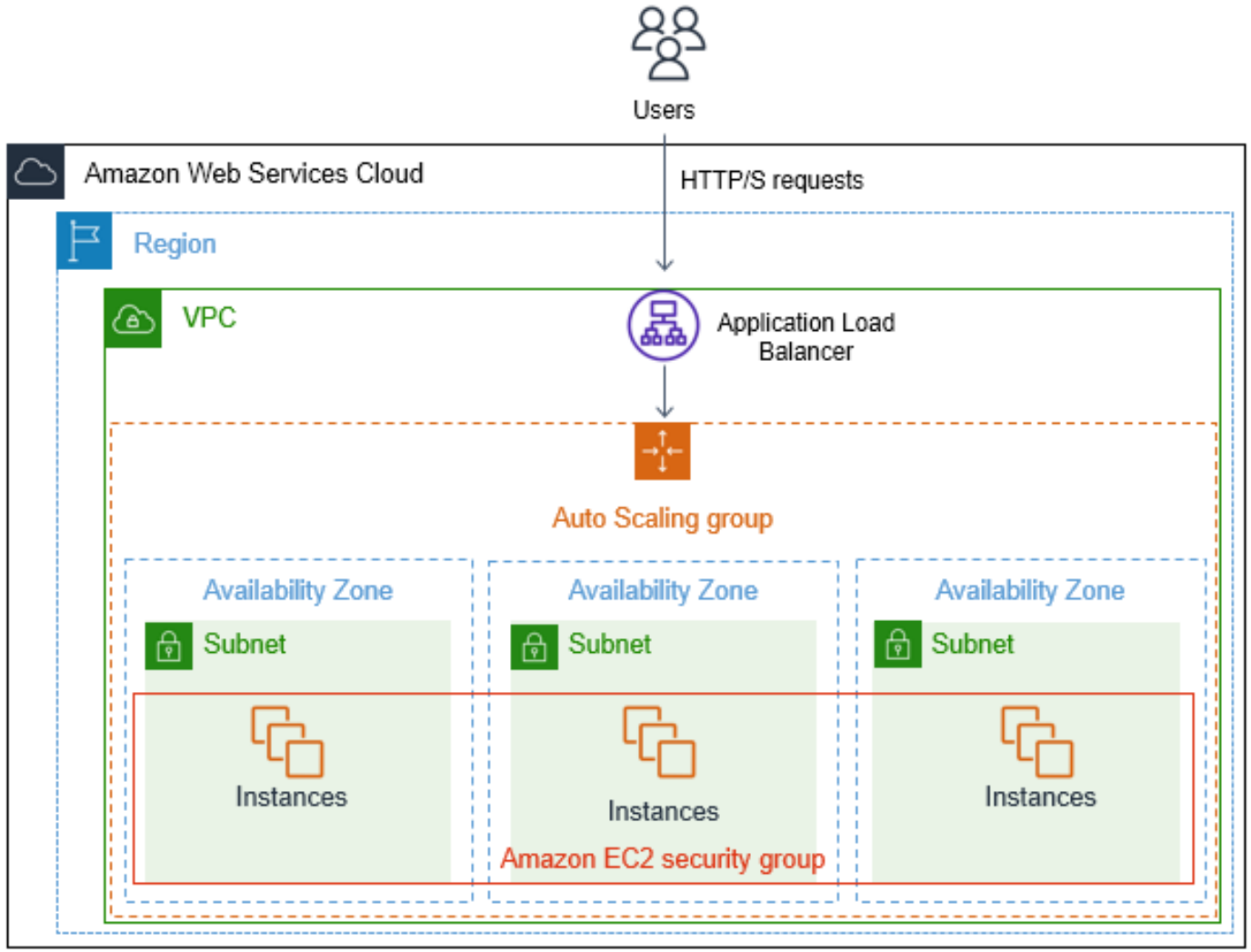
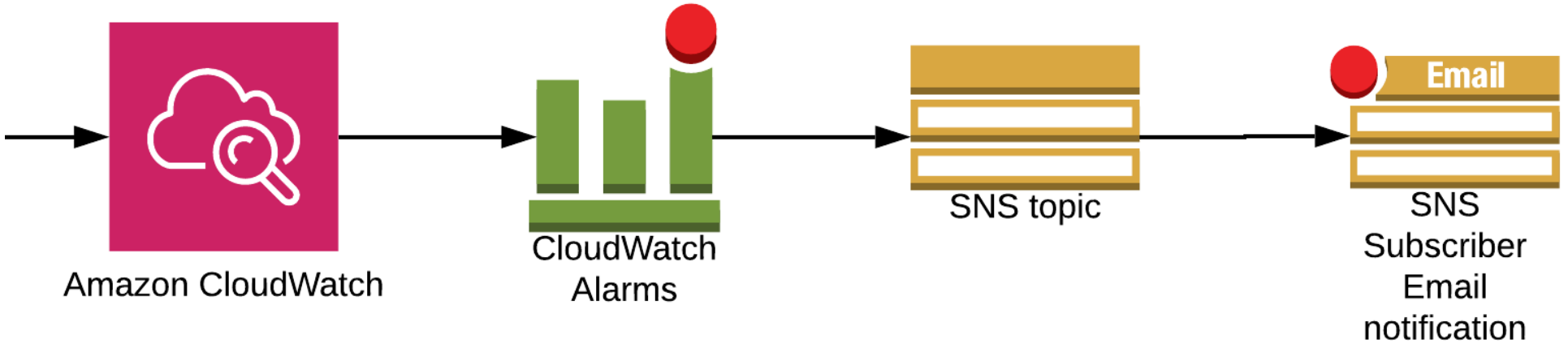


Fig: https://docs.aws.amazon.com/autoscaling/ec2/userguide/tutorial-ec2-auto-scaling-load-balancer.html

Monitoring and alerts က လည်း AWS resources တွေ layers တွေမှာ implementation လုပ်ထားသင့်တယ်။ 2/5

CPU, Memory usage က 99, 100% တောက်လျှောက်စိတ်စိတ်ဖြစ်နေတယ်။ ကိုယ်ကမသိဘူး
App ကိုခေါ်သုံးလို့တက်မလာမှ server down တာသိရင် recover လုပ်ရတာ, ဒီ server down failure ကို mitigate လုပ်ရတာ တိုင်ပတ်သွားနိုင်တယ်။

ဒါကြောင့် usage monitoring တွေ | health check တွေကို layersတိုင်း၊ resources တိုင်းမှာ လုပ်ထားသင့်တယ်။
Cloud watch ဆိုရင် AWS စလုပ်မဲ့သူဆိုလည်း မစိမ်းလောက်ဘူး။ metrics တွေ alarms တွေ default, custom လုပ်လို့ရတယ်။ threshold limit သတ်မှတ်ထားပြီး limit ကျော်လာရင် alarm တက်လာပြီး case ကို aware ဖြစ်နေရင် failure မဖြစ်အောင် mitigation လုပ်နိုင်တာပေါ့။



Cloud watch alarm တက်တိုင်း ကိုယ့်ဆီ mail ရောက်ဖို့ဆို SNS topic နဲ့ တွဲထားပြီး topic ကို subscribe လုပ်လို့ရတယ်။ cloud watch alarm event ပေါ် မူတည်ပြီး actions တွေ trigger လုပ်မယ်၊ invoke လုပ်မယ်ဆို Lambda function ပါသုံးပြီး integrate လုပ်လို့ရတဲ့ use case တွေလည်းရှိပါသေးတယ်။

Reliability အကြောင်းပြောပြီဆို ပါလာတာ how to securely deploy changes ပဲဖြစ်ပါတယ်။

3/5

ကိုယ်ရဲ့ application က new functionality, big update patching တွေလိုလာပြီဆို new changes တွေ လုပ်ရတော့မယ်၊ deploy ရမယ်ပေါ့။ For example: Users တွေသုံးနေတဲ့ current services တွေ နေရာမှာ new services အကုန် တန်းပြောင်းပေးလိုက်ဆိုပြီး အလွယ်လုပ်လို့ရတယ်။ ဒါပေမဲ့ new services တွေကို users တွေ သုံးလို့မရရင် server down ပါပြီ၊ fail ပါပြီပေါ့။ New services တွေနေရာမှာ အရင်ဟာတွေ ပြန်ပြောင်းဖို့ကလည်း လွယ်လွယ်ကူကူ roll back မနိုင်ဘူးဆို recover time ကြာသွားပြီး lack of reliability ဖြစ်နိုင်ပါတယ်။ မုန့်ဟင်းခါးစားရပါပြီ။

အဲ့လိုတွေမဖြစ်အောင် securely deploy changes လုပ်ဖို့ဆို deployment pattern တွေ သုံးပြီး controlled changes တွေနဲ့ လုပ်လို့ရတယ်။ Controlled changes လို့ပြောရတာက for example. Current infra နဲ့ New infra ဆိုပြီး ဆောက်ထားတယ်။ User တွေက current infra က application ရဲ့ services တွေကို သုံးနေတယ်။ New infra က services နှစ်ခုလောက်ကို current infra ရဲ့ services နှစ်ခုနေရာမှာ users တွေသုံးလို့ရအောင် စမ်းပို့ကြည့်လိုက်တယ်။ အဆင်ပြေရင် နောက်ထပ် services တွေထပ်ပို့မယ်။ မပြေရင် ချက်ချင်း current infra က services တွေပဲ ပြန်သုံးလို့ရအောင် roll back ပြန်လာမယ်ဆိုတာမျိုးပေါ့။ အလွယ်ဆုံးရှင်းပြလိုက်တာမို့ပါ။ Deployment pattern တွေက canary deployment, blue/green, feature flags ဆိုပြီး အသေးစိတ်ရှိပါတယ် ။

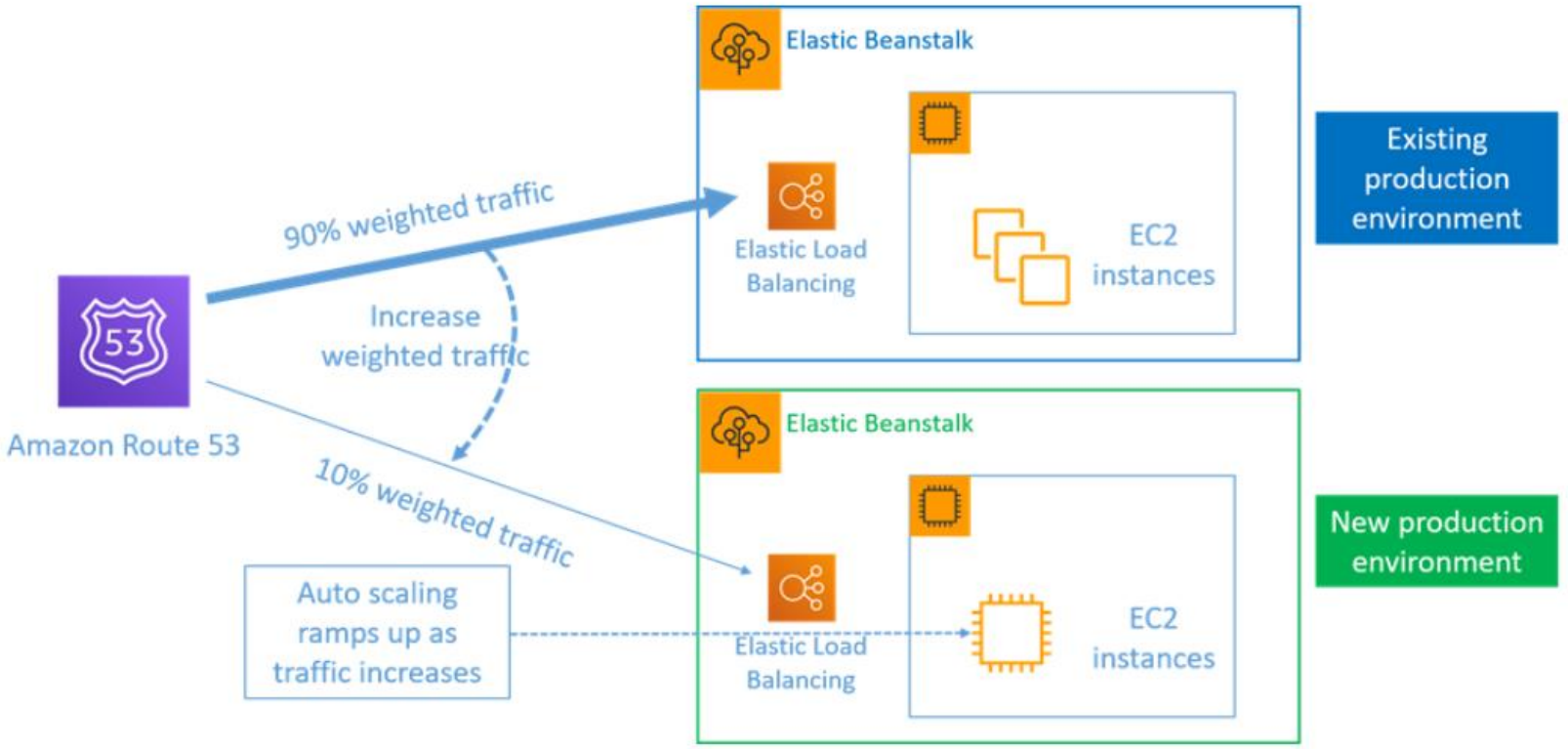


Figure: Blue/green deployment with AWS Elastic Beanstalk and Amazon Route 53

To securely deploy changes: https://docs.aws.amazon.com/wellarchitected/latest/reliability-pillar/implement-change.html

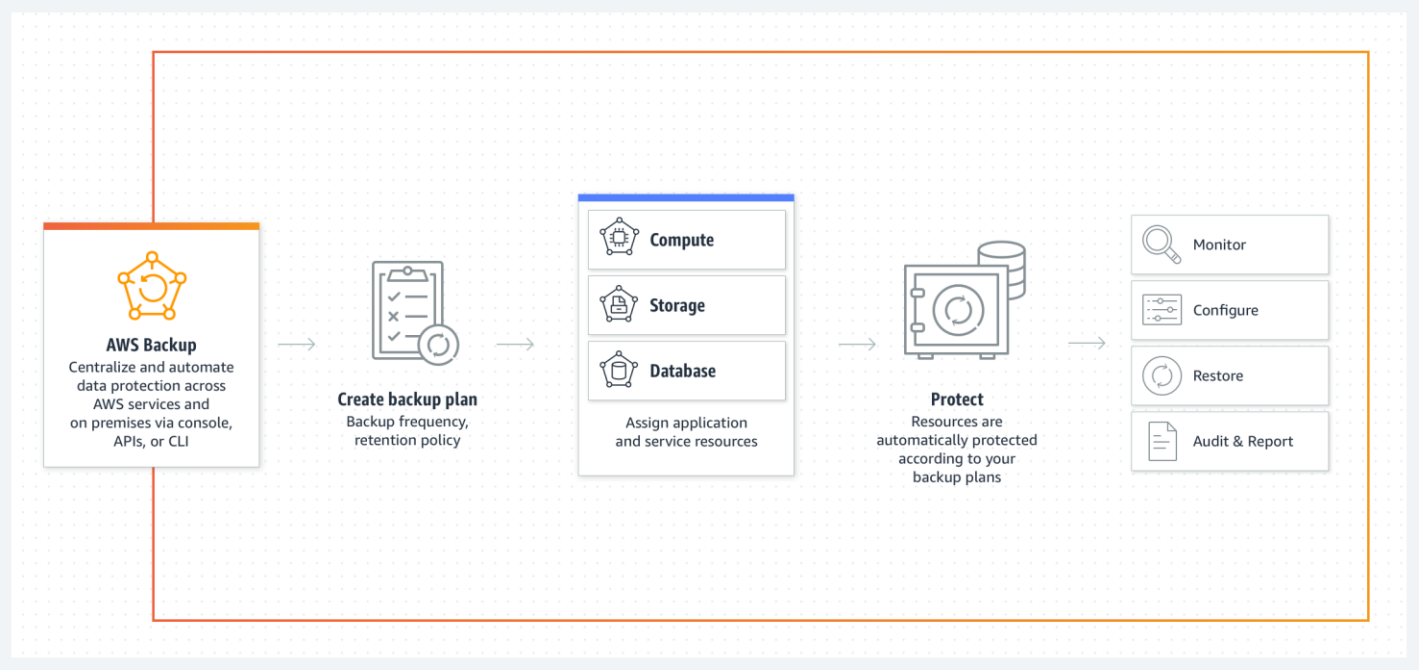
Backup တွေကလည်း Cloud resources တွေရဲ့ reliability အပေါ် impact ထဲထဲဝင်ဝင်ရှိပါတယ်။

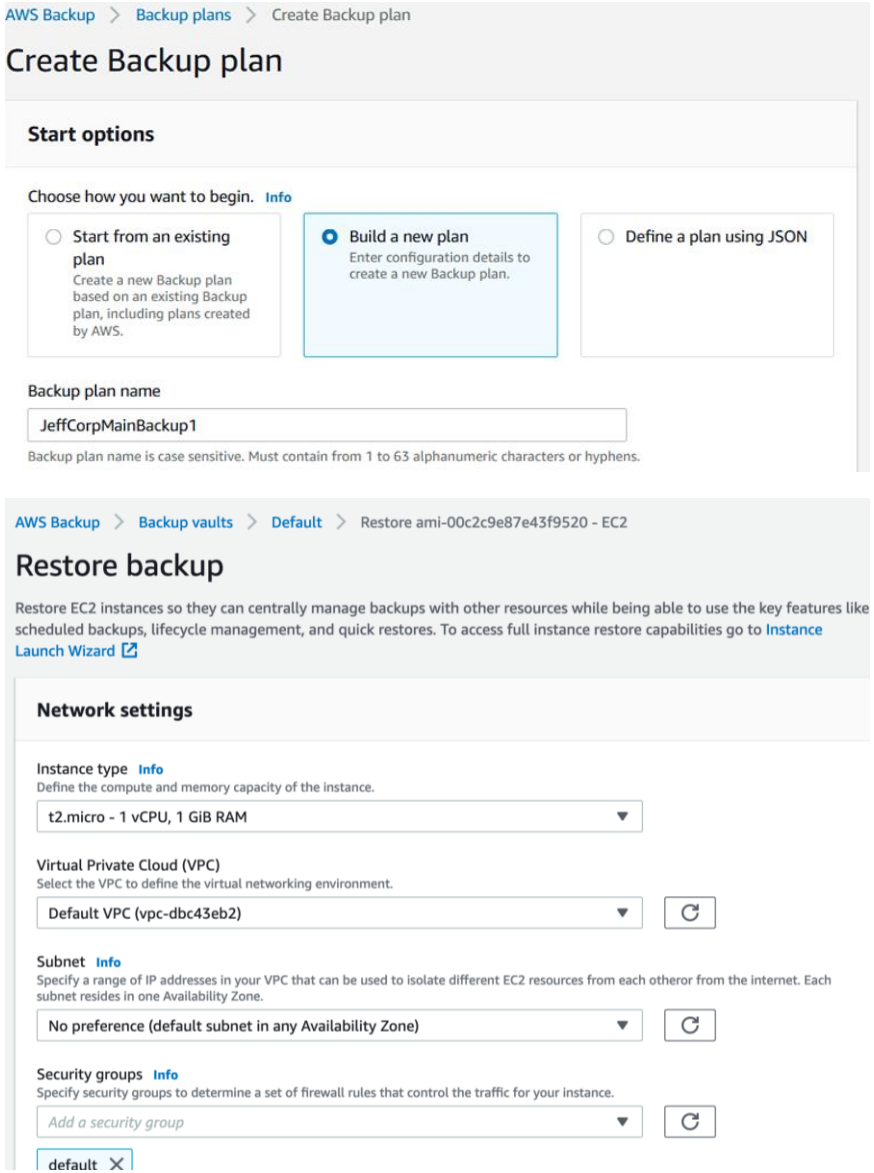
4/5

AWS resources တွေက လွယ်လွယ်ကူကူ backup ယူပြီး သိမ်းထားလို့ရပါတယ်။ back size က ကြီးပြီး သိမ်းထားမဲ့ အချိန်က ကြာမယ်ဆို cost ကတော့ တက်နိုင်တာတော့ ဂရုစိုက် ရမှာပေါ့။ surprise cost memes တွေက အတည်ဖြစ်လာမယ် :3

Backup snapshot ယူထားပြီး လိုအပ်တဲ့ အချိန်မှာ restore လို့ရတာက reliability အတွက်ကောင်းပါတယ်။ data AWS မှာက Backup service ဆိုပြီးရှိတယ်။ Backup plan တွေ automated schedule တွေ custom ရေးပြီး backup သိမ်းချင်တဲ့ resources တွေ assign ထည့်ထားလို့ရတယ်။ သိမ်းထားမဲ့အချိန် retention period ကိုလည်း days, weeks, months သတ်မှတ်လို့ရတာပေါ့။ ပုံမှန်တော့ daily backup ယူတာကို 2 weeks retention၊ monthly backup ယူတာကို 6 months retention ထားတာကောင်းတယ်။ Hourly backup ကတော့ ယူလို့ရလားဆိုရတယ်။

for example: 50GB EBS volume ကို hourly backup snapshot တစ်ပတ်လုံးယူပြီး တစ်ပတ် retention သိမ်းမယ်ဆို 50Gb x 24hours x 7 days x 0.05$ = 420$ ဆိုပြီး အကြမ်းဖြင်းတွက်လို့ရပါတယ်။ Use case ကတော့ data loss sensitive ဖြစ်တဲ့ အချိန်တွေမှာပဲ hourly backup လုပ်သင့်တယ်။ cost concern က surprise တိုက်နိုင်တာ ဂရုစိုက်ရပါမယ် ☺။





AWS Backup : https://aws.amazon.com/backup/

Reliability pillar မှာ backup အကြောင်းပါပြီးဆိုတော့ သူနဲ့ တွဲနေတဲ့ Disaster recovery လဲလေ့လာရပါမယ်။

5/5

DC, DR ? Data center sites နဲ့ Disaster recovery sites ဆိုပြီး သိကြမှာပါ။
Disaster recovery ဘာလို့သုံးရတာလဲအရင်ပြောကြတာပေါ့။ သဘာဝဘေးအန္တရာယ်တွေကြောင့် On premise Infra တစ်ခုလုံး fail တာမျိုးတို့၊ AWS region လိုက် AZ လိုက် data center တွေ failတာမျိုးတို့ ဖြစ်လာဖို့ခက်ပေမဲ့ business giant company တွေက reliability အတွက် မူလdata center sites တွေ fail သွားရင် တစ်ခြား disaster recovery sites တွေ business critical services တွေ data တွေ migrate ဖို့ DC/DR solution ကိုသုံးကြတယ်။

DC,DR solution အတွက်ကတော့ AWS ပေါ်မှာ AWS Elastic Disaster Recovery (DRS) service ဆိုပြီး လွယ်လွယ်ကူကူသုံးလို့ရတာရှိတယ်။ ကိုယ် DR ထားချင်တဲ့ on premise ကပဲဖြစ်ဖြစ်၊ cloud ပေါ်ကပဲဖြစ်ဖြစ် DC resources တွေကို console ပေါ်ကနေပဲ manage လုပ်ပြီး DR set up လုပ်နိုင်ပါတယ်။
Recovery pattern တွေကတော့ infra ထပ်တူဆောက်ပြီး မူလ DC fail တာနဲ့ တန်းပြောင်းတဲ့ multi-site(active/active) အပြင် warm standby, pilot light, backup & restore ဆိုတာတွေလေ့လာလို့ရတယ်။
Disaster recovery solution တွေက infra ထပ်တူဆောက်ပြီး recover တဲ့ recover pattern မူကွဲလေးတွေပဲ ဆိုတော့ DC 1 ကျပ်၊ DR 1 ကျပ် ကုန်တာတော့ aware ရမှာပေါ့ 😉

Application တစ်ခုကို EC2 server နှစ်ခုက runနေရင် AZ မတူအောင်ထားတာတို့၊ RDS မှာဆိုလည်း multi-AZ deployment တို့ Aurora database မှာဆို read replica ကို တစ်ခြား region မှာဆောက်ထားပြီး main database fail ရင် read replica ကို stand by အနေနဲ့ primary DB အဖြစ် setup တဲ့ best practices တွေ ရှိပါတယ် ခဗျ။

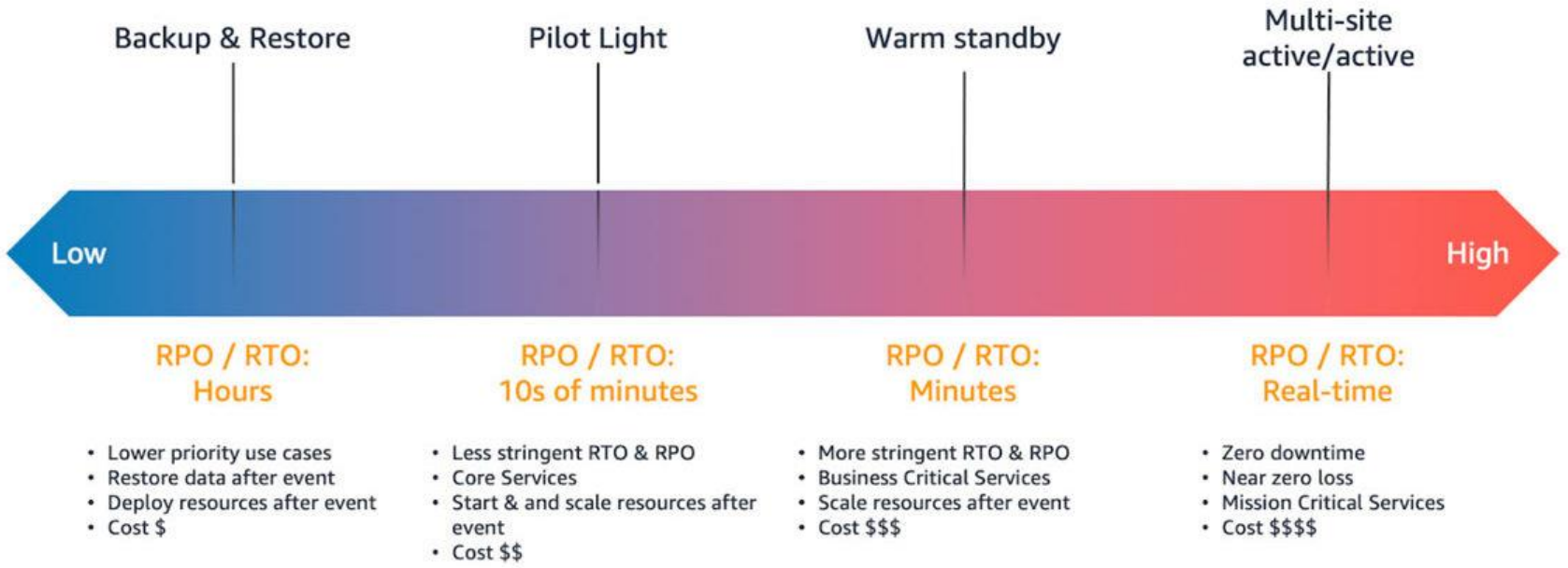


Fig: https://docs.aws.amazon.com/whitepapers/latest/disaster-recovery-workloads-on-aws/disaster-recovery-options-in-the-cloud.html

Thank you