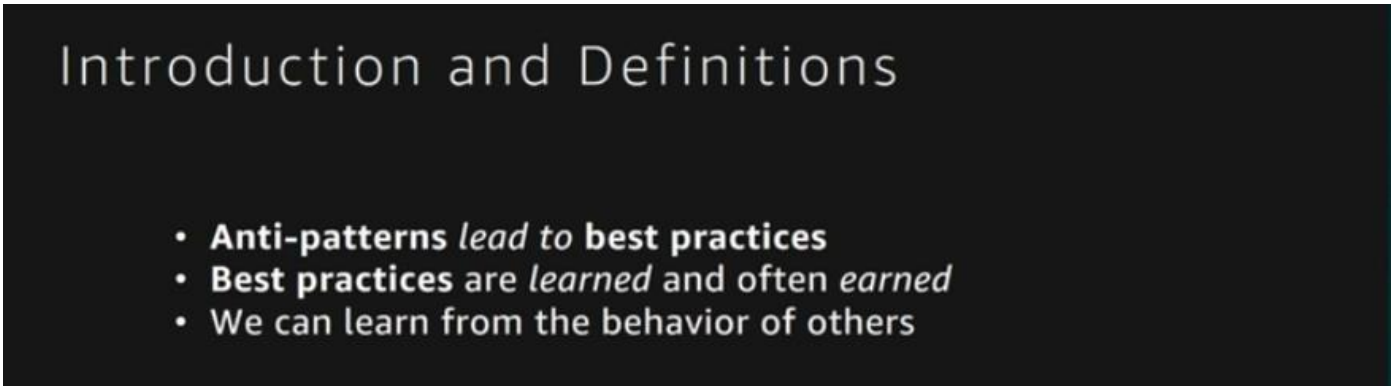
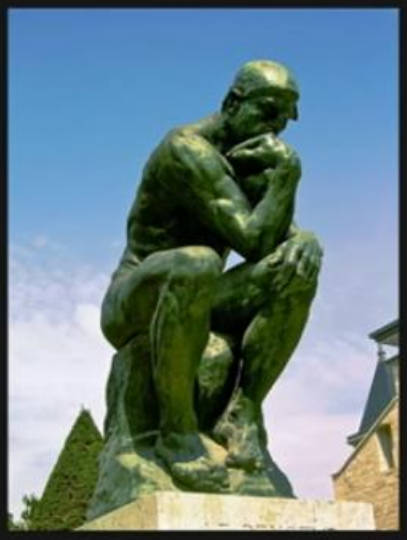
How does a practice become a "best" practice? How does a pattern become an "anti" pattern? As always, experience is the best teacher. As Partner Solution Architects, we receive a lot of partner feedback on how practices and design patterns work—and occasionally fail to work—in the real world. We use this feedback to inform our recommendations and reference architectures. In this session, we explore a representative set of real-life "failures." We look at what these failures have to teach us about design and how to prioritize remediation of known issues.



Best Practice Creation—Myth



This work has been released into the public domain by its author, AndrewHorne at English Wikipedia. This applies worldwide.

AWS re:Invent

aws

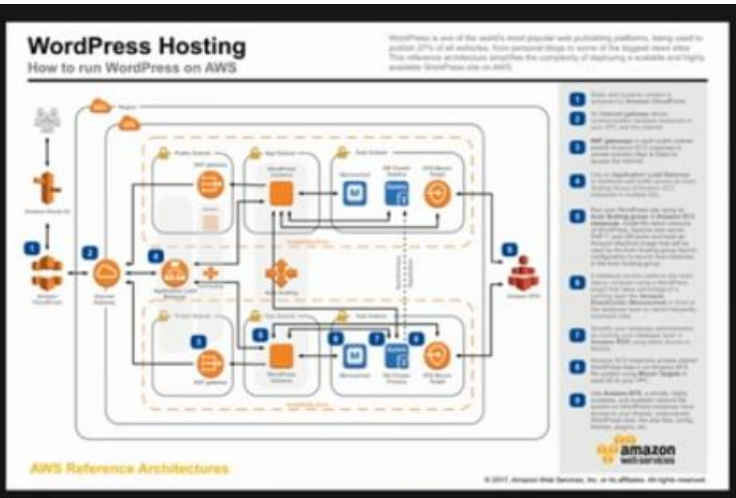
Best Practice Creation—Reality

By Sylvain Pedneault - Self-photographed, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=3616567

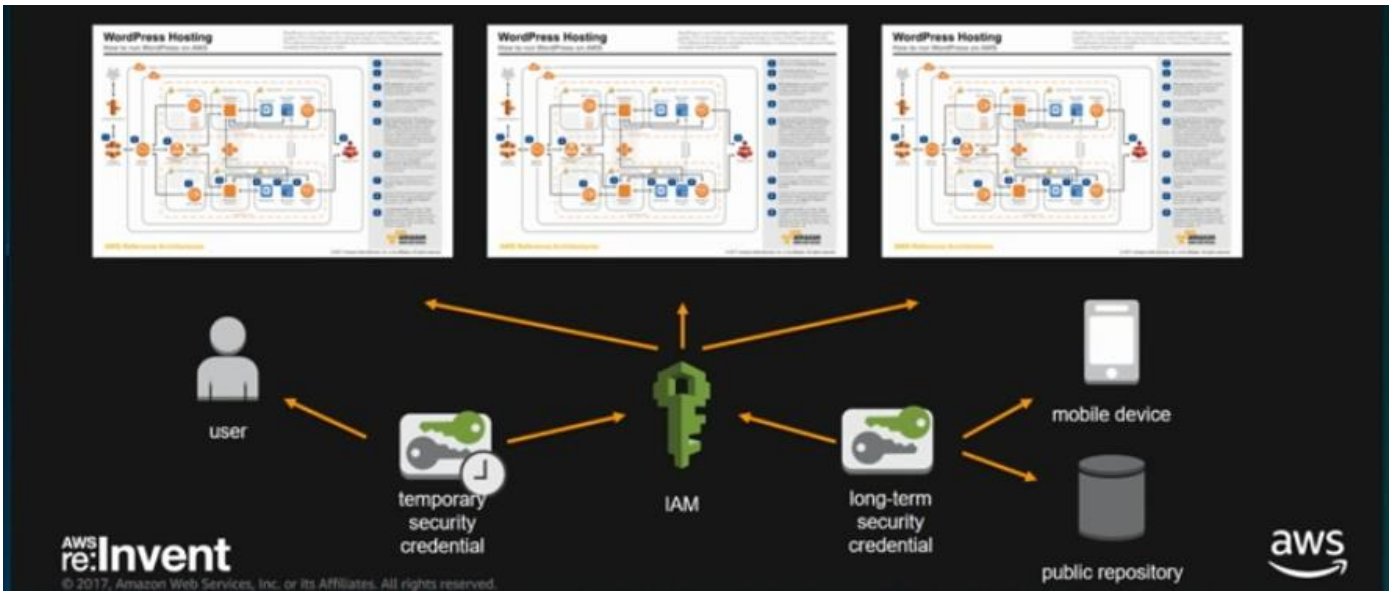
AWS re:Invent

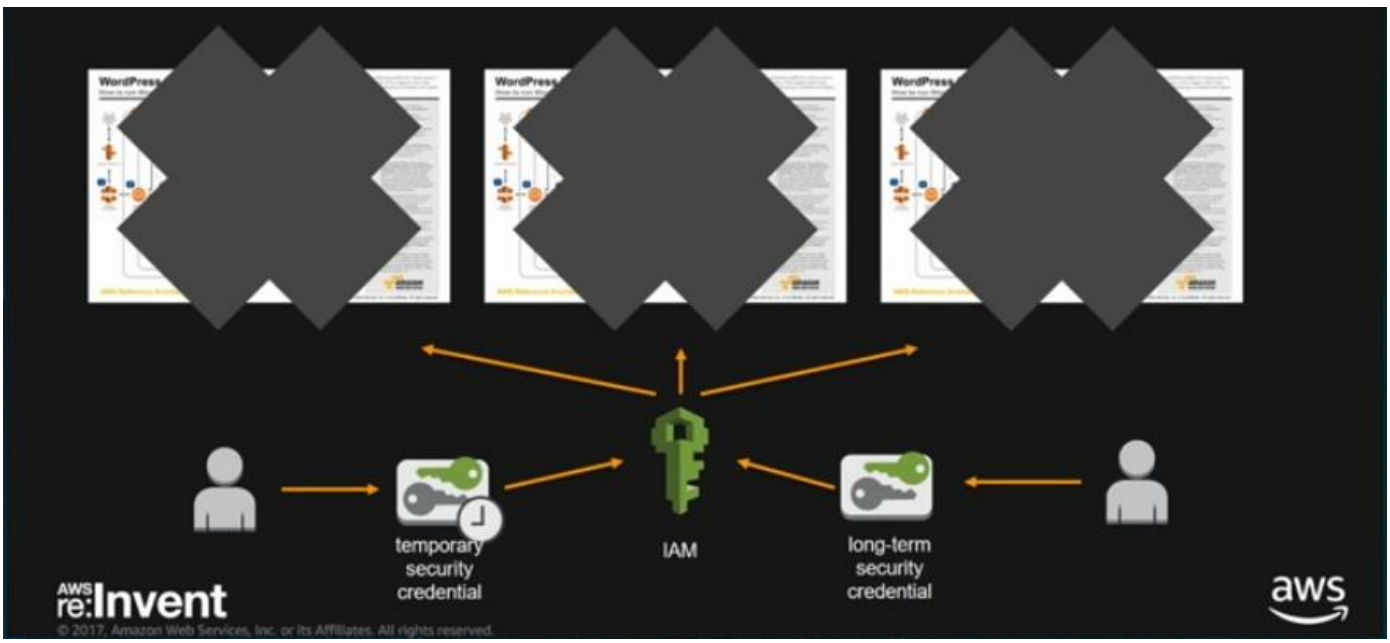
aws

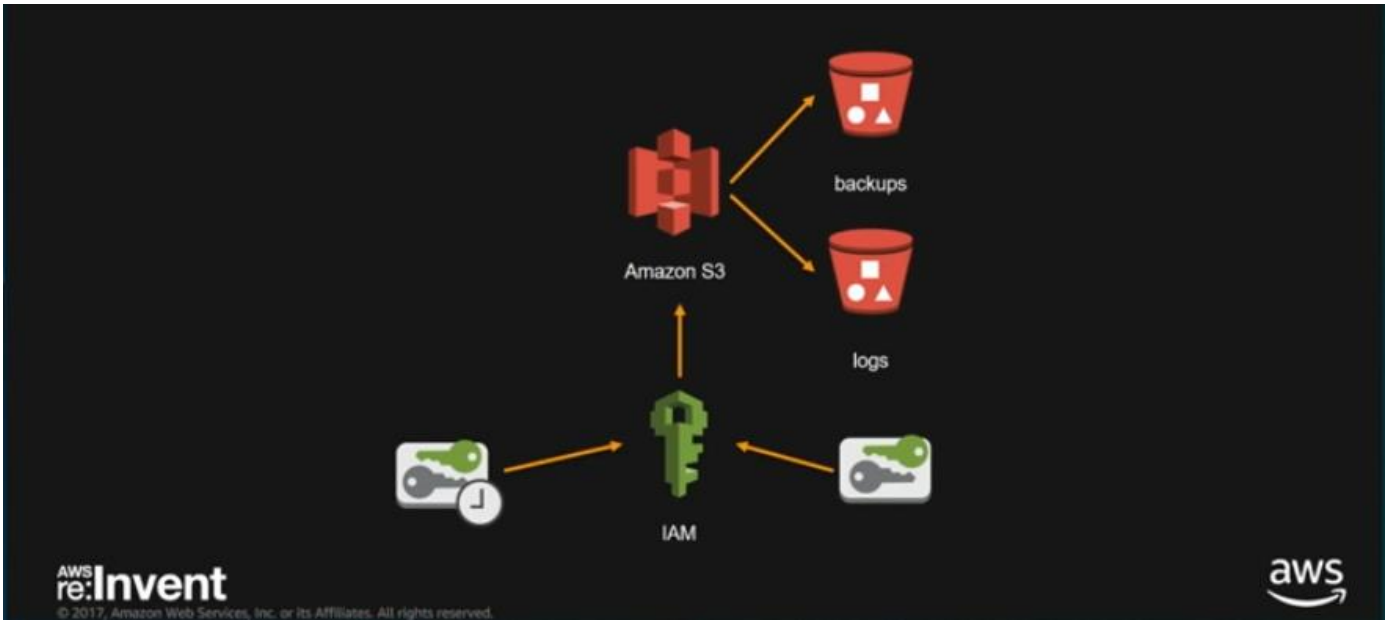
Anti-Pattern: Loss of Control

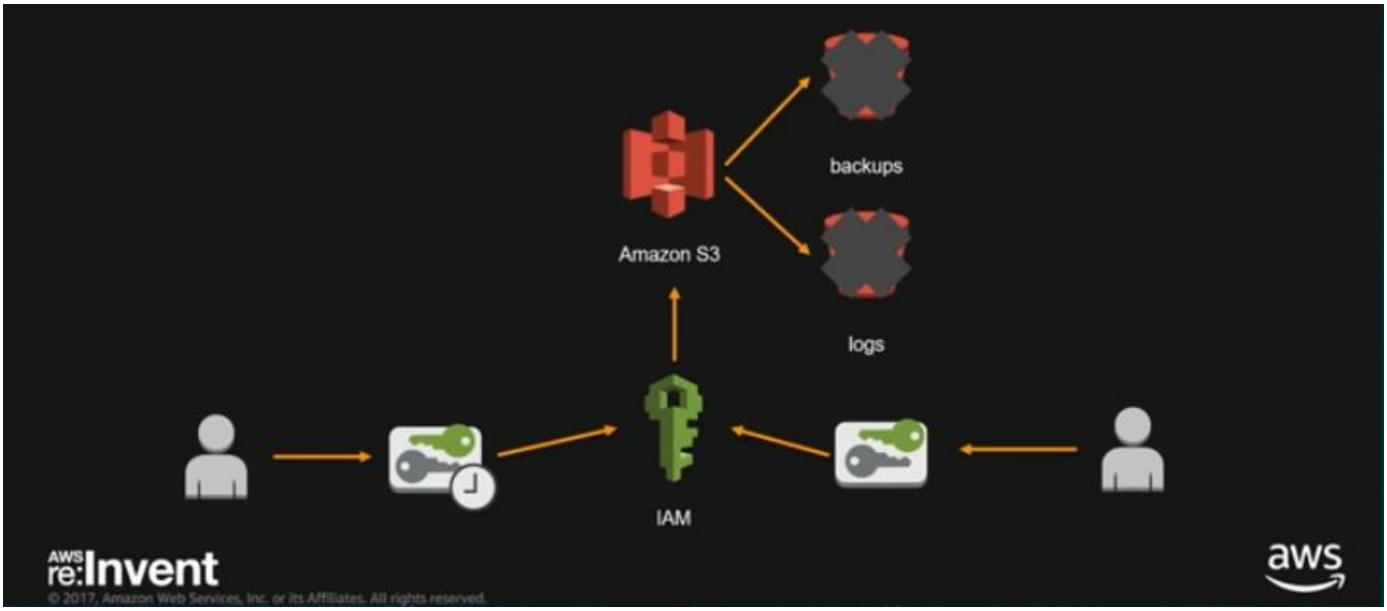


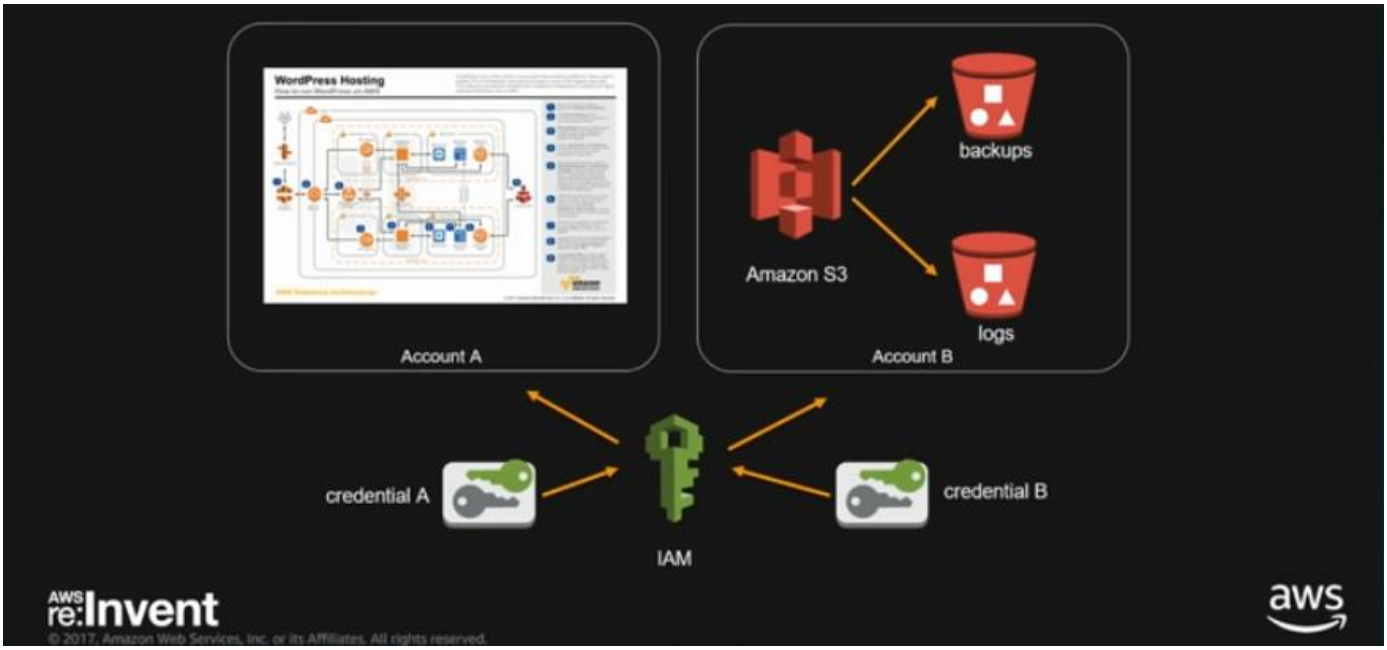
https://github.com/awslabs/aws-refarch-wordpress

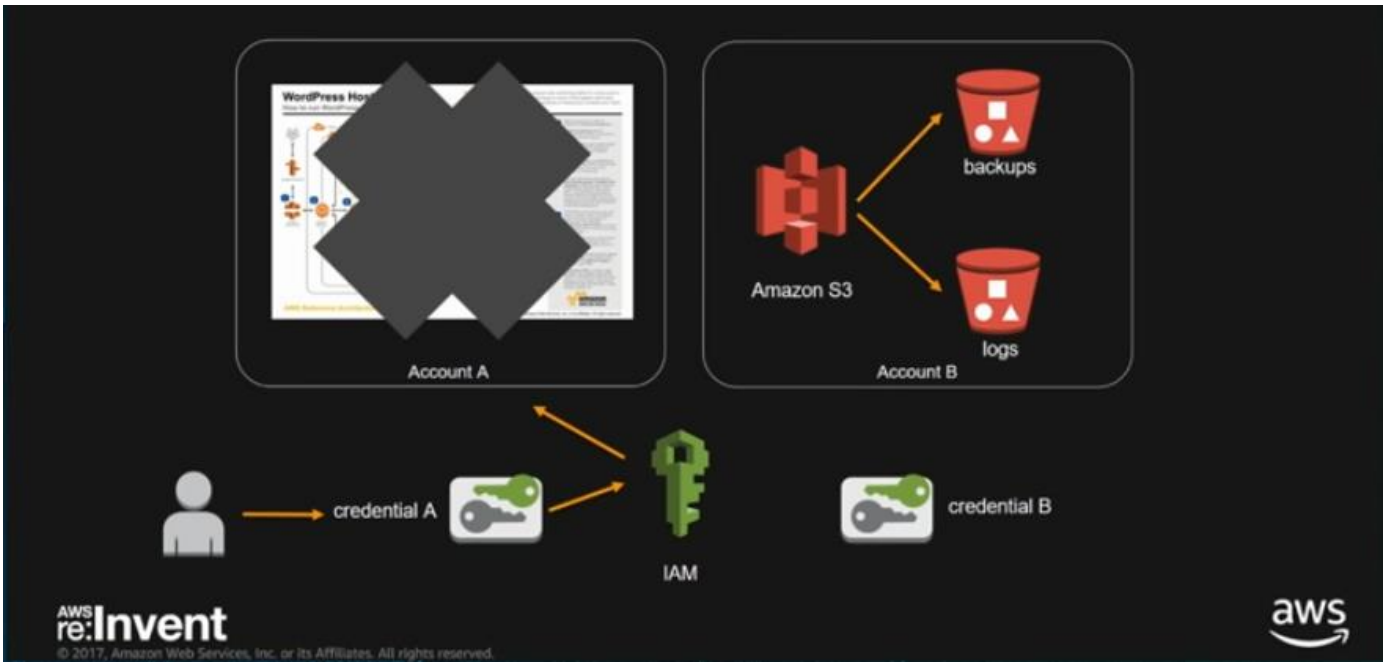


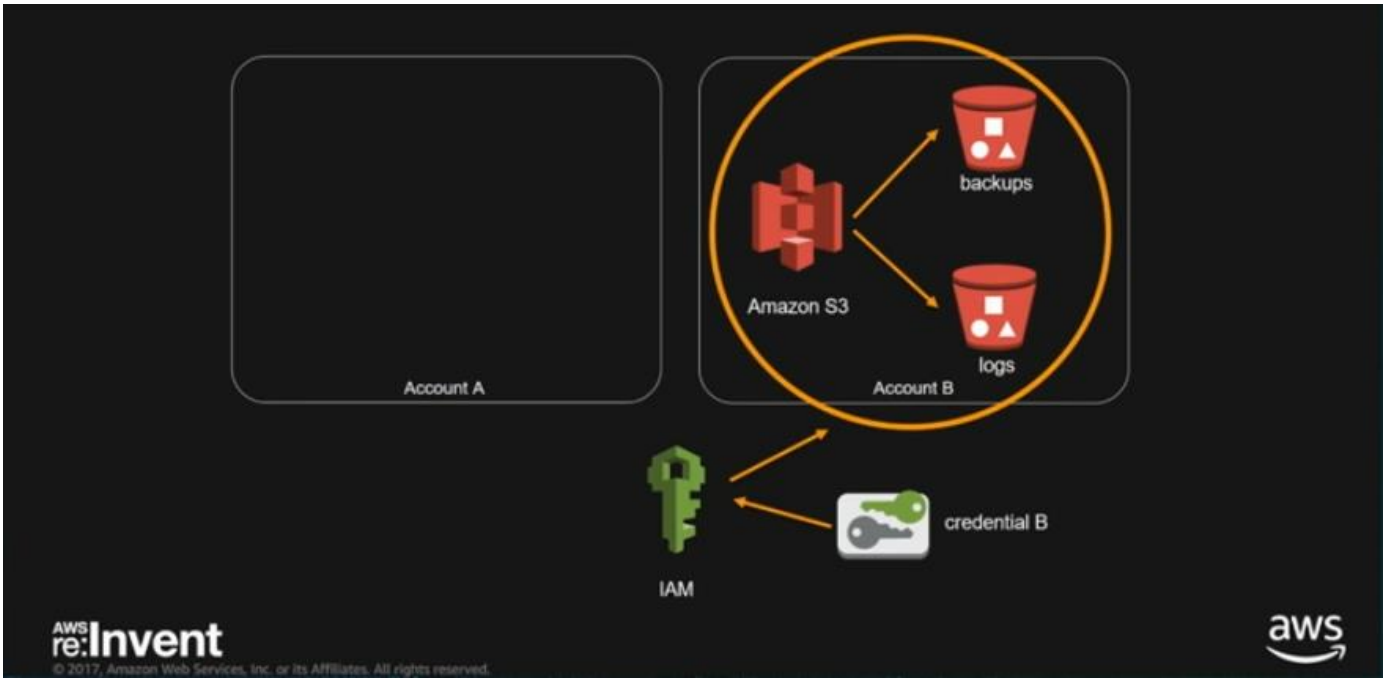


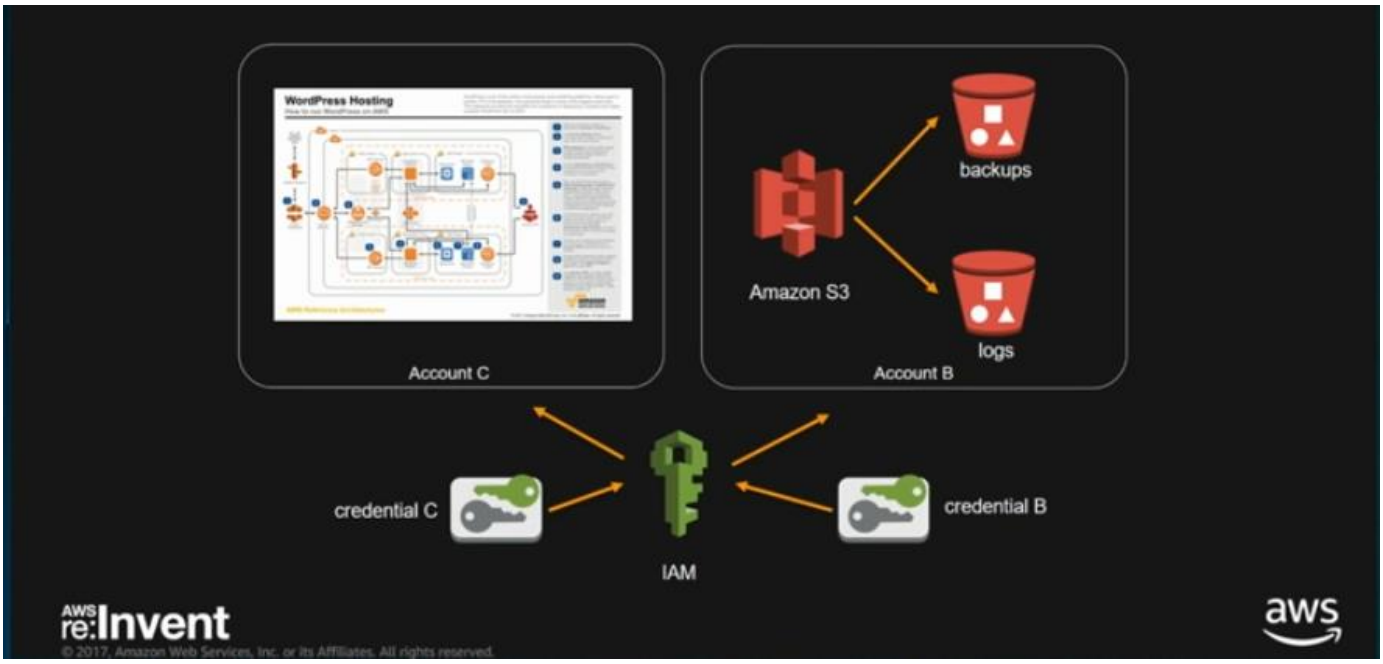

backups
Amazon S3
logs
IAM
aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws


backups
Amazon S3
logs
IAM
aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws


WordPress Hosting
Account A
backups
Amazon S3
logs
Account B
credential A
credential B
IAM
aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws


Account A
backups
Amazon S3
logs
Account B
credential A
credential B
IAM
aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws


Account A
backups
Amazon S3
logs
Account B
credential B
IAM
aws re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws



Anti-Pattern: Loss of Control

Anti-pattern: **Poor IAM Access Key controls**

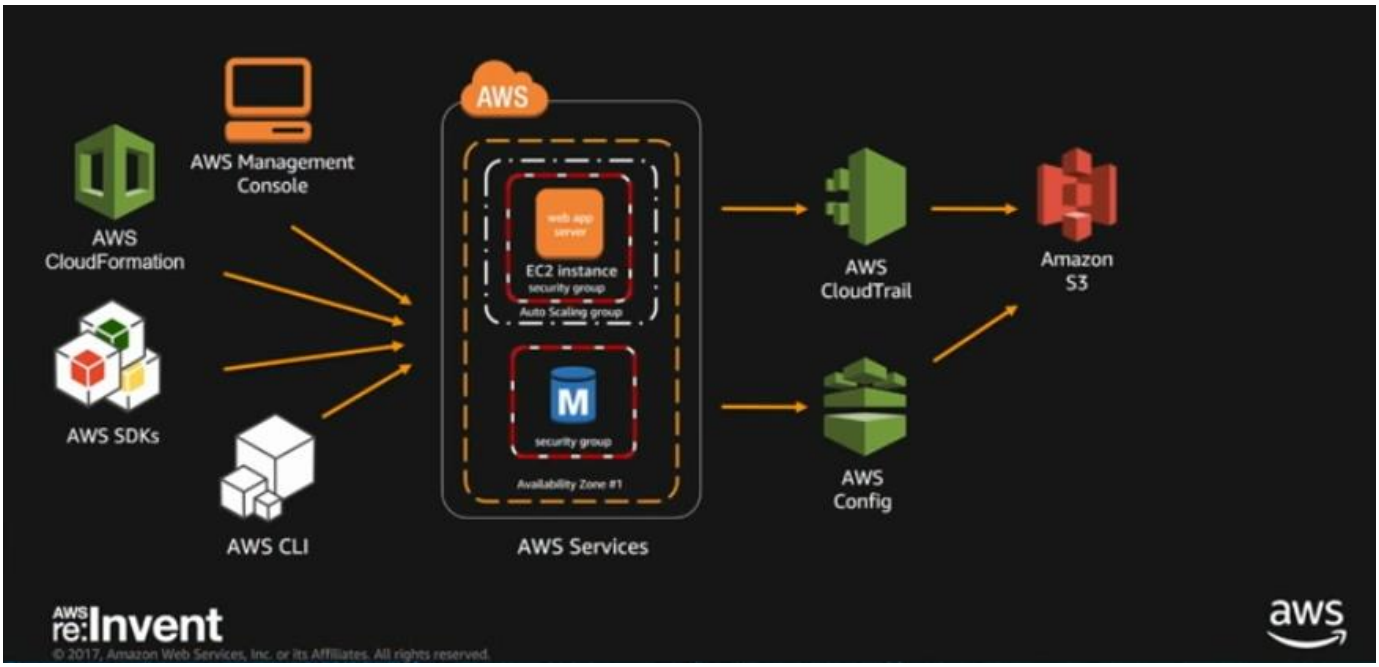
Best practices:

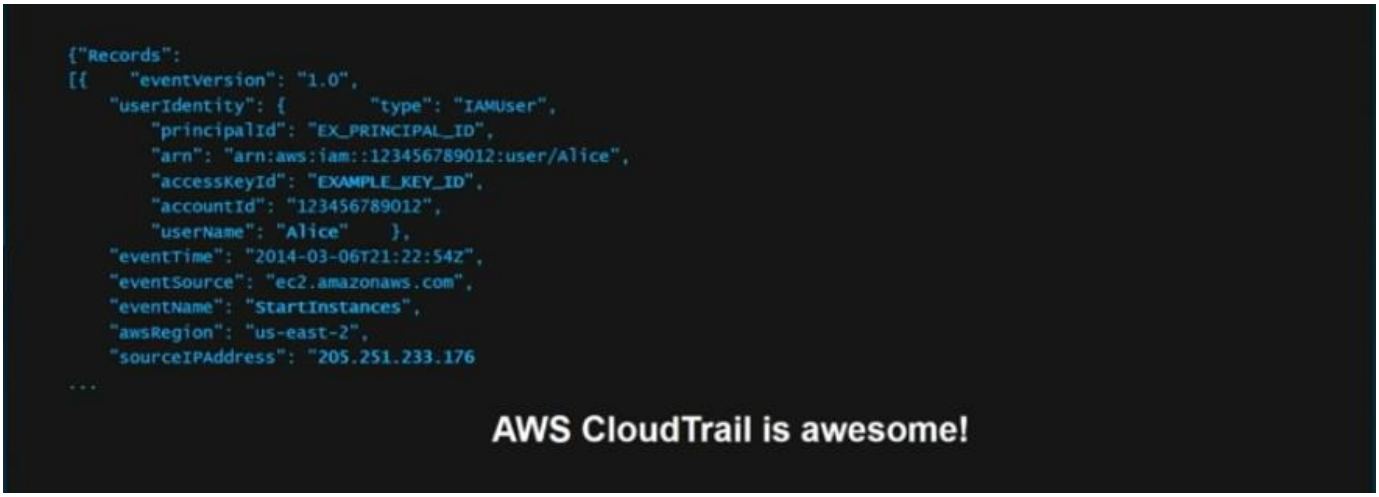
1. Lock away your AWS account root user access keys
2. Create individual IAM users
3. Enable MFA for privileged users
4. Never automate with privileged credentials
5. Rotate credentials regularly
6. Audit for compliance
7. Establish separate administrative domains

..and regularly review access policies with an AWS Solutions Architect!

More AWS IAM Best Practices
http://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Anti-Pattern: Control Gaps





AWS CloudTrail gives us logs of everything that goes on in our account

AWS Config gives us a point-in-time snapshot of the inventory of the assets in our accounts

```
AWSTemplateFormatVersion:2010-09-09
Resources:
  SGBase:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Whitelist Security Group
      SecurityGroupIngress:
        - IpProtocol: tcp
        CidrIp: 167.55.180.10/0
        FromPort: '22'
        ToPort: '22'
```

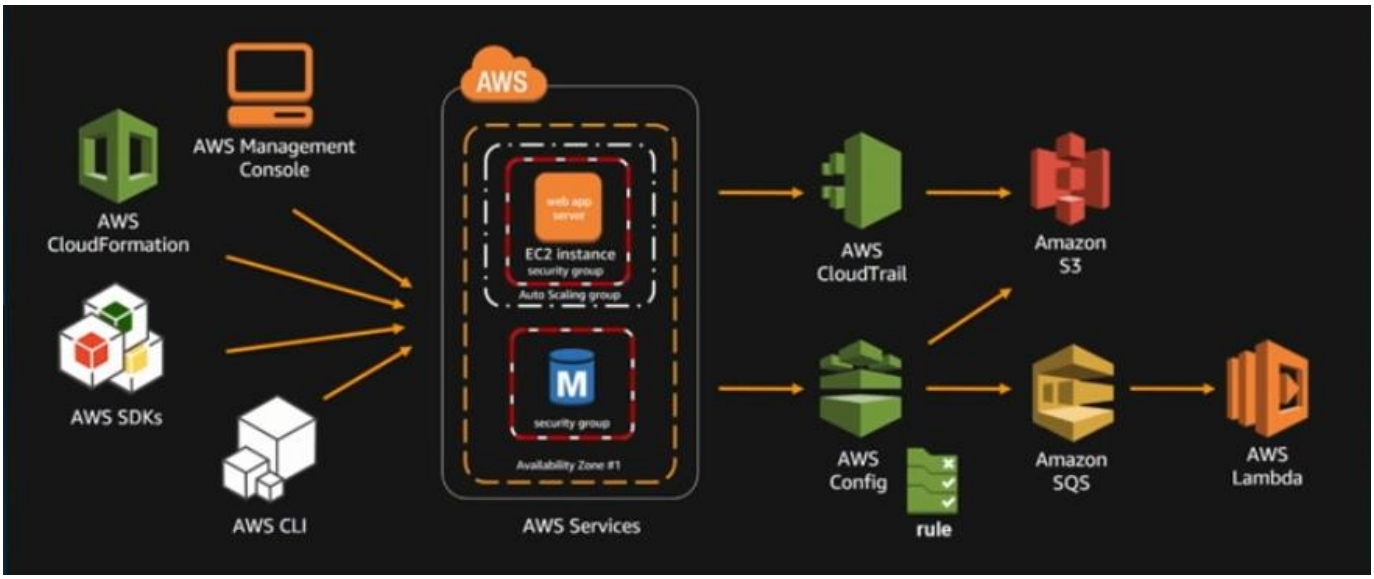
What's wrong with this picture?

```
AWSTemplateFormatVersion:2010-09-09
Resources:
  SGBase:
    Type: 'AWS::EC2::SecurityGroup'
    Properties:
      GroupDescription: Whitelist Security Group
      SecurityGroupIngress:
        - IpProtocol: any
        CidrIp: 167.55.180.10/32
        FromPort: '3388'
        ToPort: '3390'
```

What's wrong with this picture?

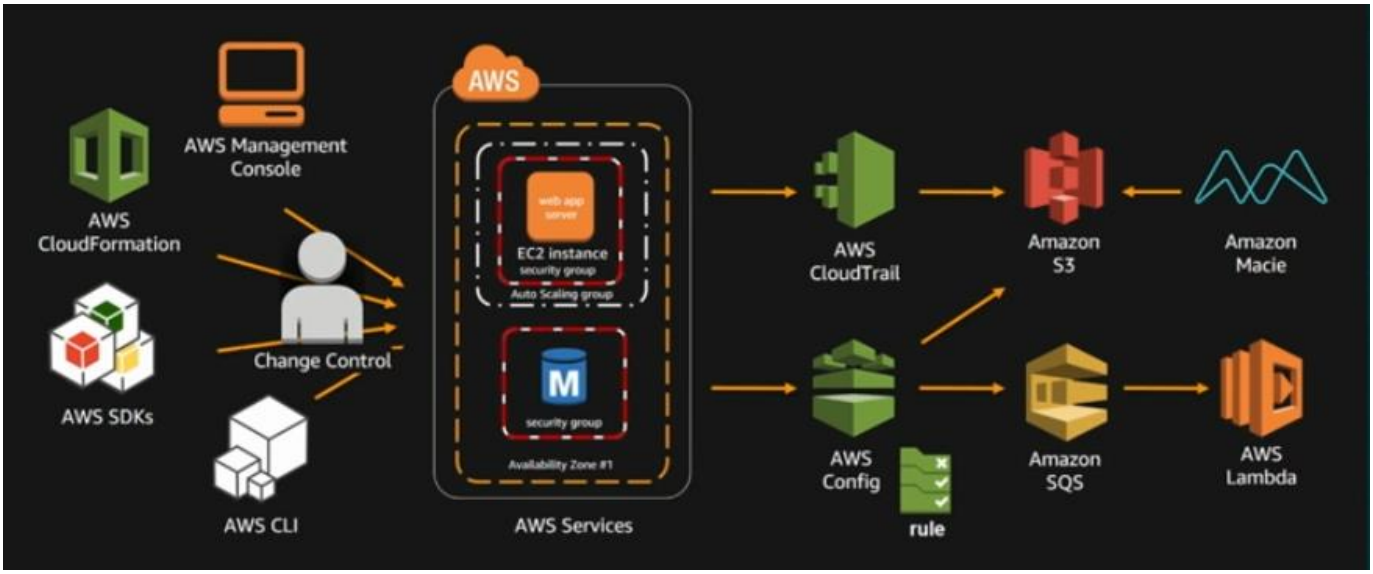
This is a bad rule to allow access that escapes automation detection but will be easily detected by a human

Amazon S3 authorization process

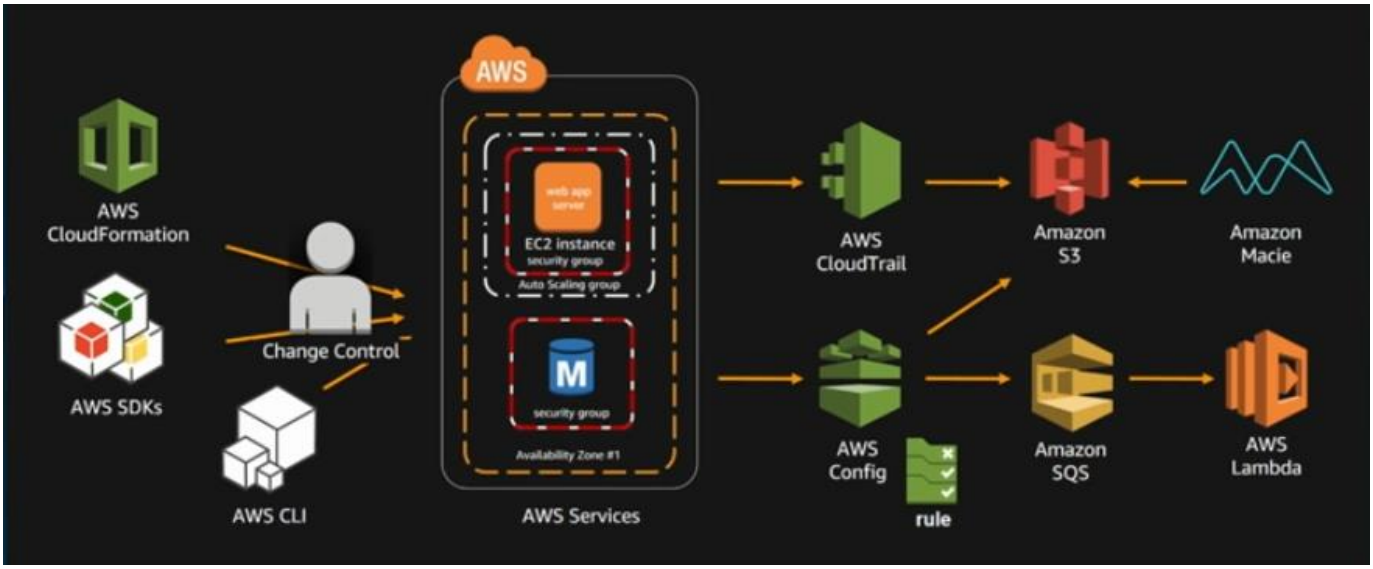


Use the AWS Managed Rules service to help close some of the control gaps in your infrastructure, also use services like SNS and Lambda to auto-correct the control gaps.

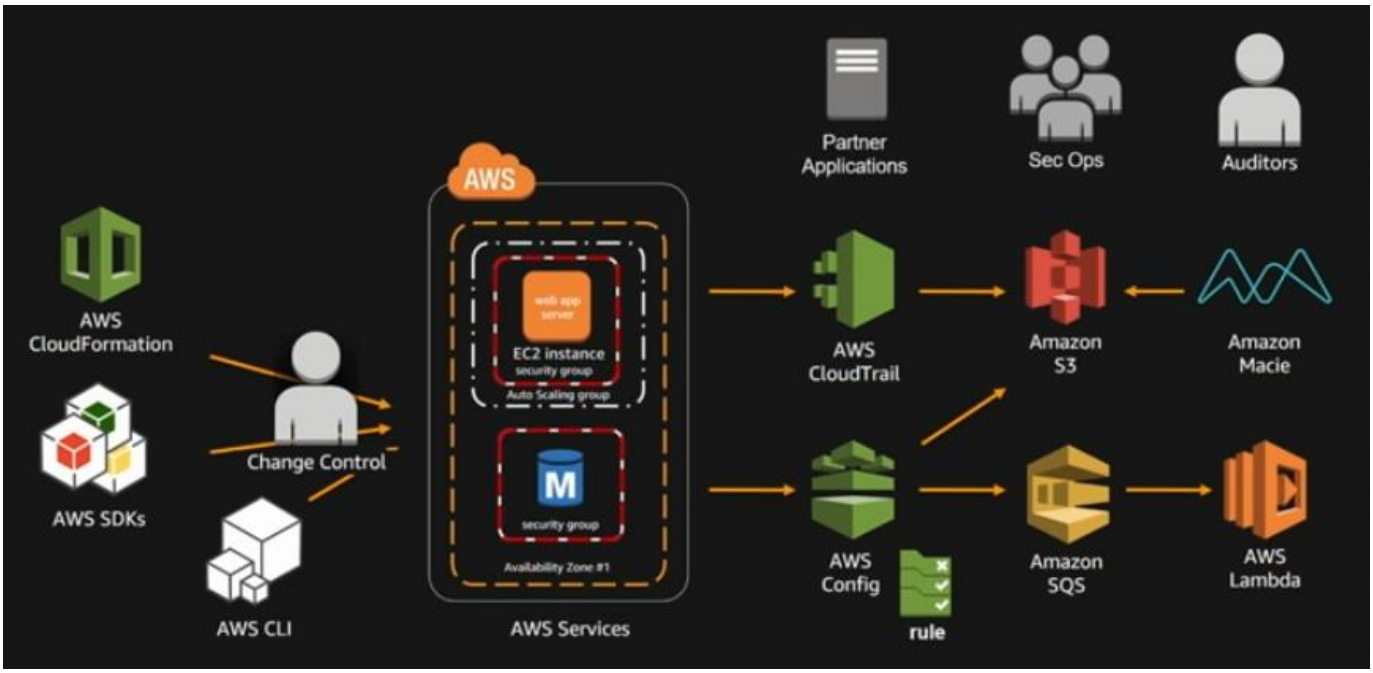
You can also use a service like Amazon Macie



Also consider using change control by using a human to look at the changes before deploying code changes



Maybe don’t use the AWS Management Console for managing changes manually, instead use automation



Anti-Pattern: Automated Control Gaps

Anti-pattern: **Reliance on incomplete controls automation**

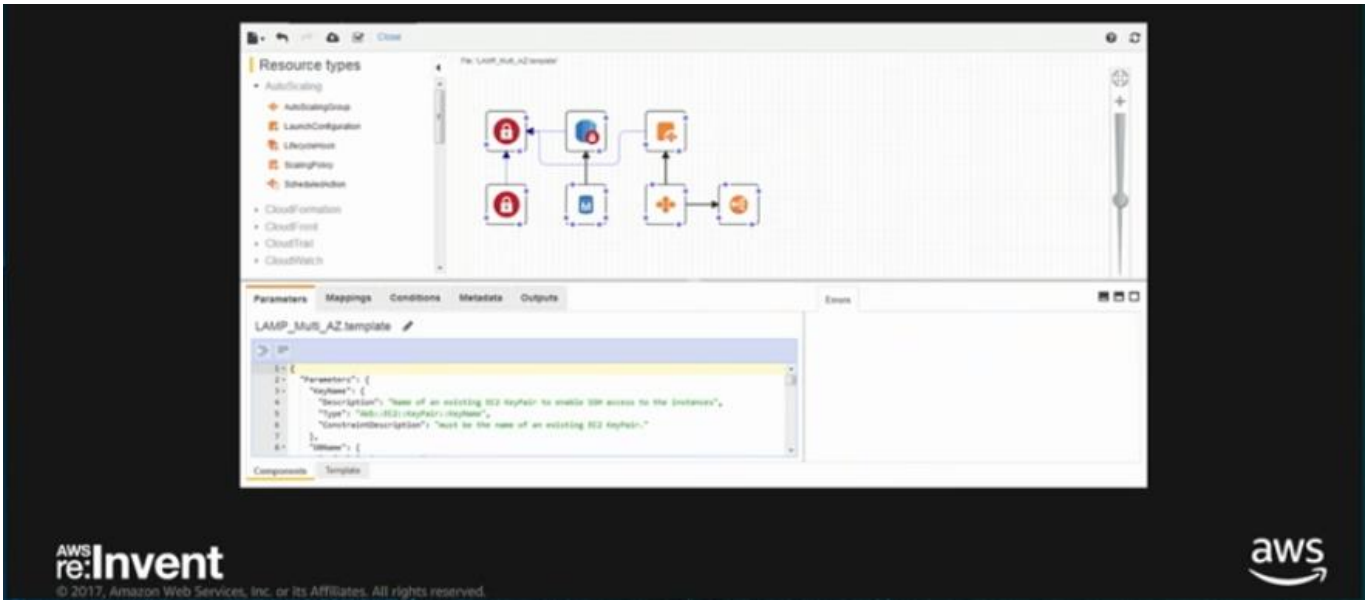
Best practices:

1. Use managed rules
2. Inject canary events to test controls
3. Use external tests and tools for validation
4. Audit to verify compliance
5. Add manual checkpoints prior to pushing changes
6. Automate everything, but mind the gaps!

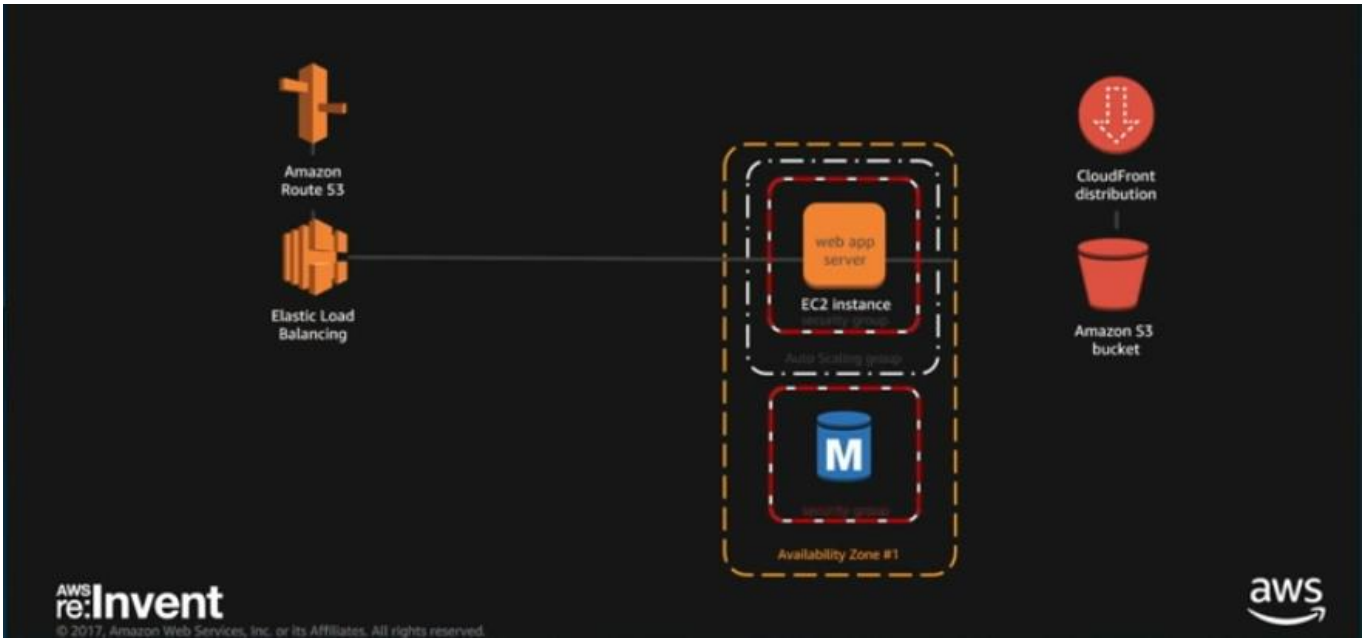
More on AWS Config Managed Rules
https://aws.amazon.com/blogs/aws/aws-config-update-new-managed-rules-to-secure-s3-buckets/

More on Automating Governance on AWS
https://www.youtube.com/watch?v=9g0u_05WBig

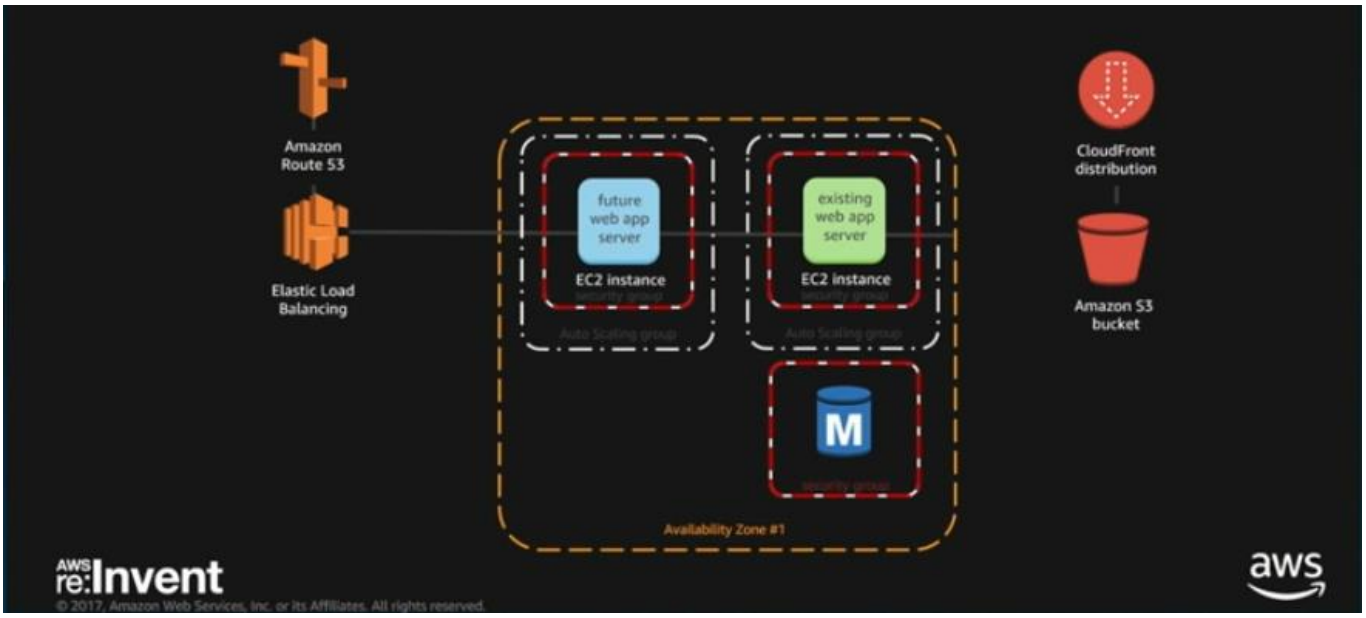
Anti-Pattern: Automating Outages



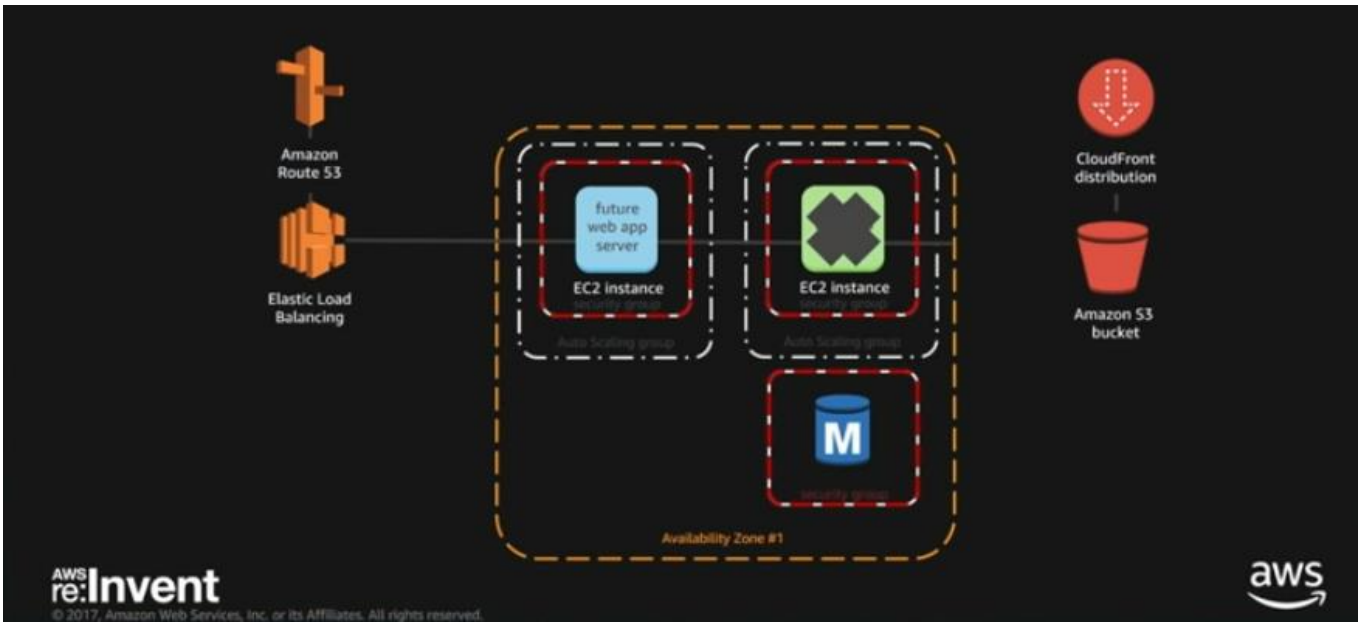
We can easily automate our deployments by using CloudFormation or similar tooling like Terraform, Puppet, etc.



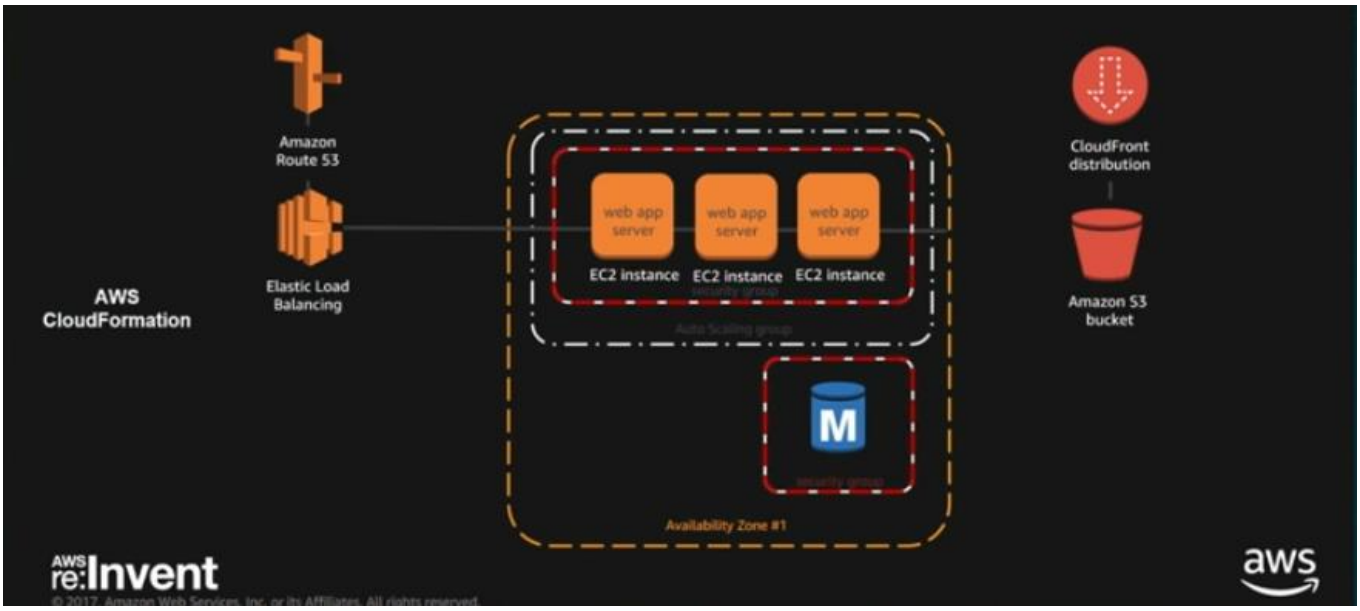
We can scale in or scale out automatically using auto-scaling groups using AWS native services as above.



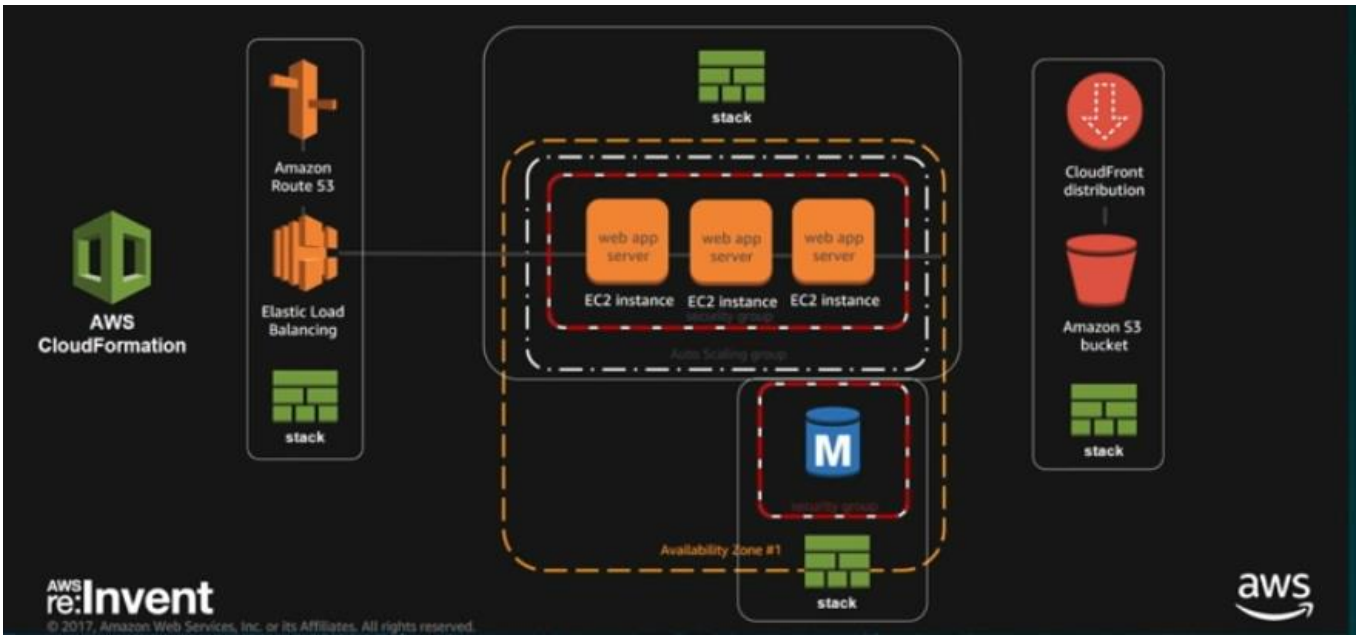
We can start to use automation and blue green deployments as above,



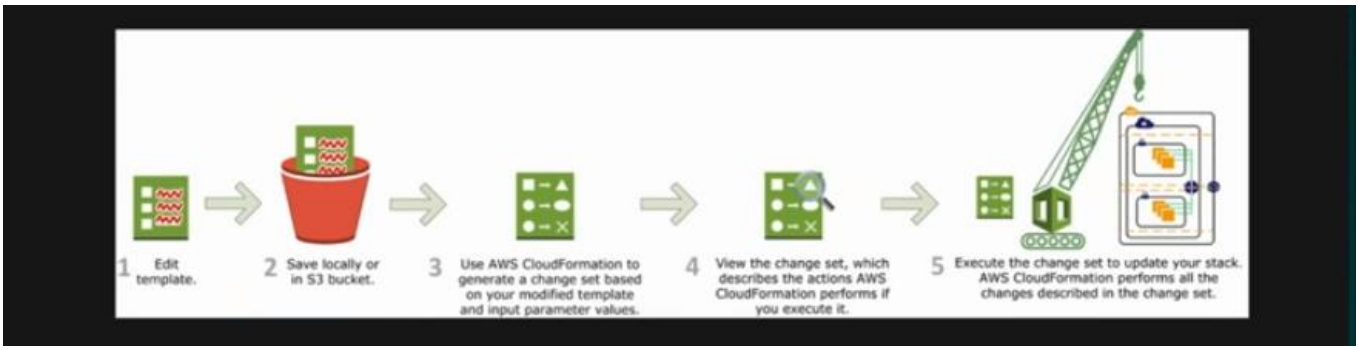
You then flip the traffic to the new stack



We can use auto-scaling group to scale our infrastructure in an automated fashion



We should be decoupling the infrastructure into stacks that are based on the responsibility of each stack application and the requirements. We have a web-server stack, database stack, CloudFront distribution stack,



This is how we can use CF to create different stacks

Anti-Pattern: Automating Outages

Anti-pattern: **Incomplete Automation and Testing**

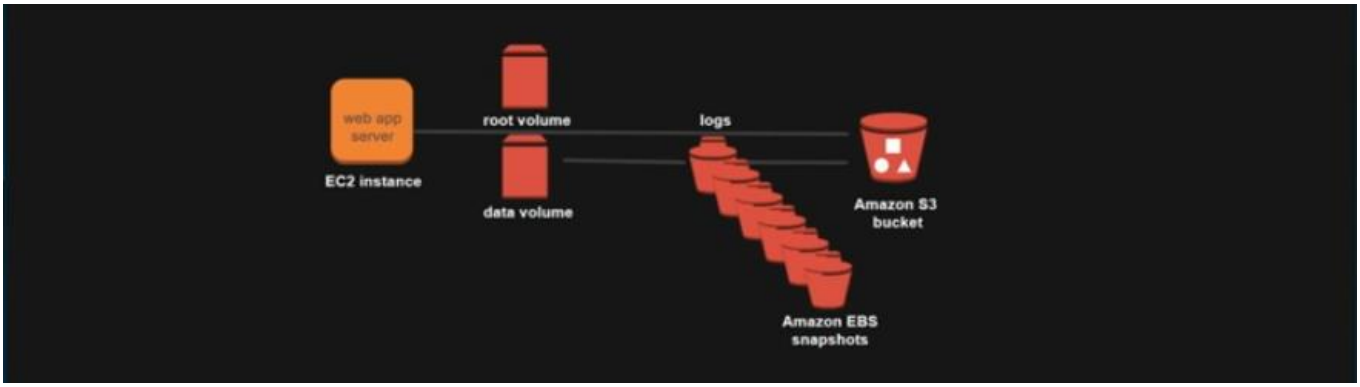
Best practices:

1. Decouple stateful and stateless infrastructure management automation
2. Limit interactive access to infrastructure
3. Define and enforce tagging policy
4. Implement blue/green and rolling upgrades
5. Test infrastructure automation in non-production environments
6. Administrative domains!

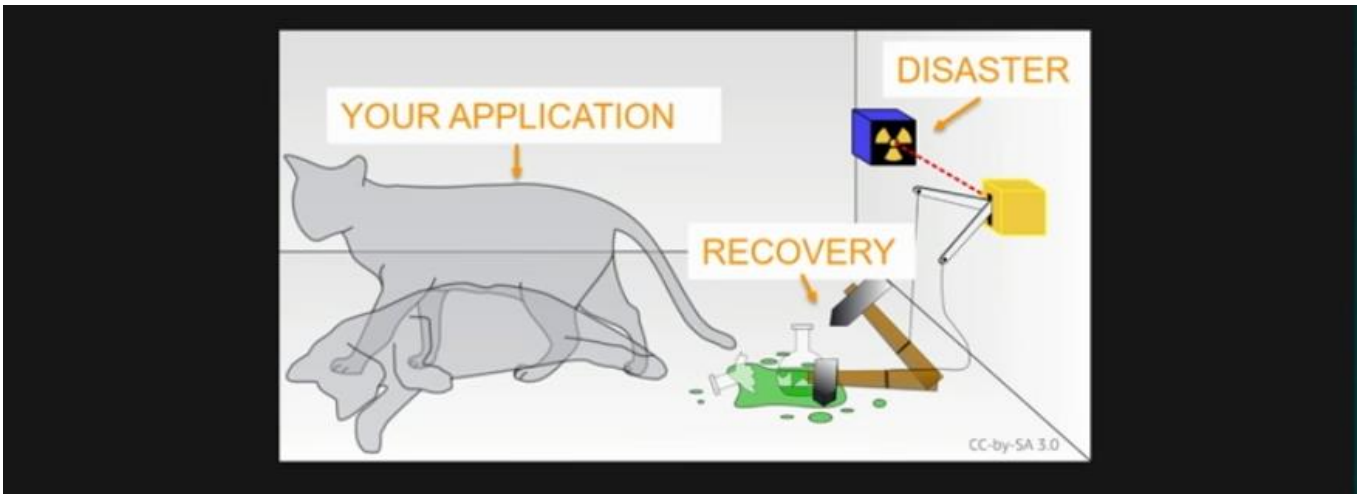
More AWS Infrastructure Automation Best Practices
http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/best-practices.html

Anti-Pattern: Schrödinger's Backup

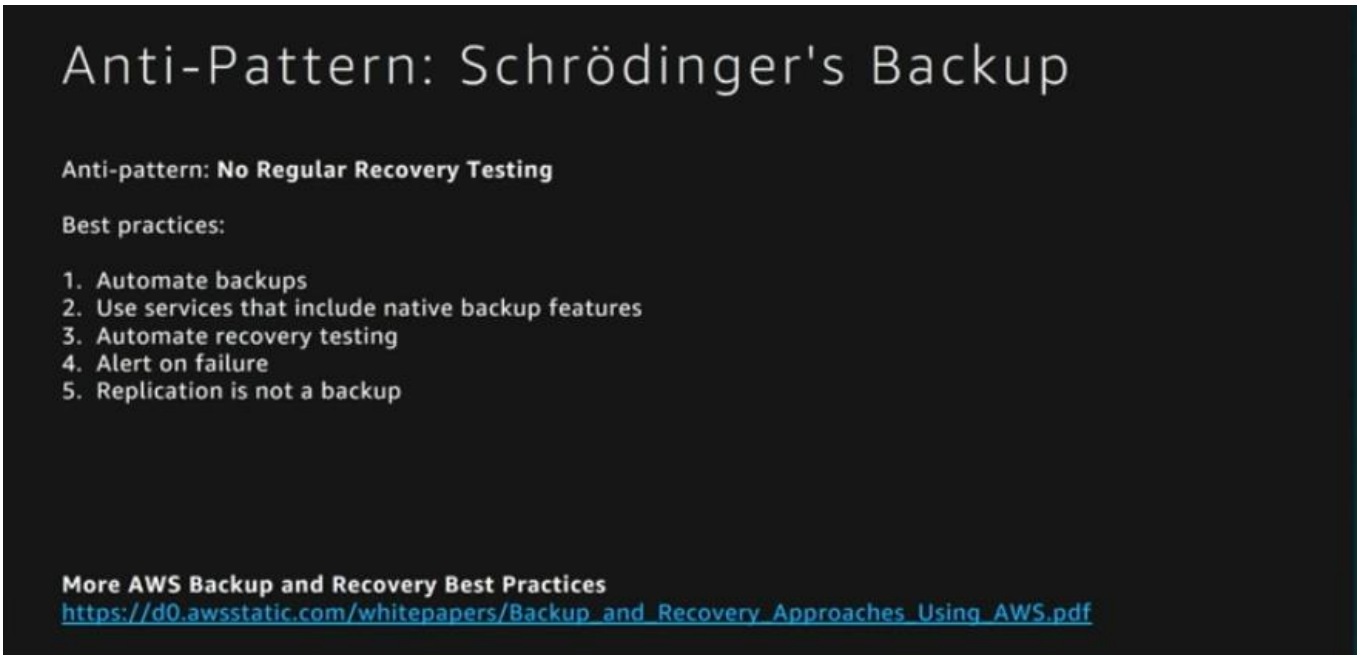
We have a sample infrastructure shown above that we want keep backed up using EBS snapshots



We then create a bunch of snapshots over time for possible recovery in an outage disaster



You need to test your backups quickly to make sure you are really covered when you need to use the backups. A test could be to make sure your backups are getting bigger in storage from yesterday to today? are you trimming older backups like 2 weeks ago? are your backups 0 KB in size? You don't need to redeploy your backups



Establishing Best Practices

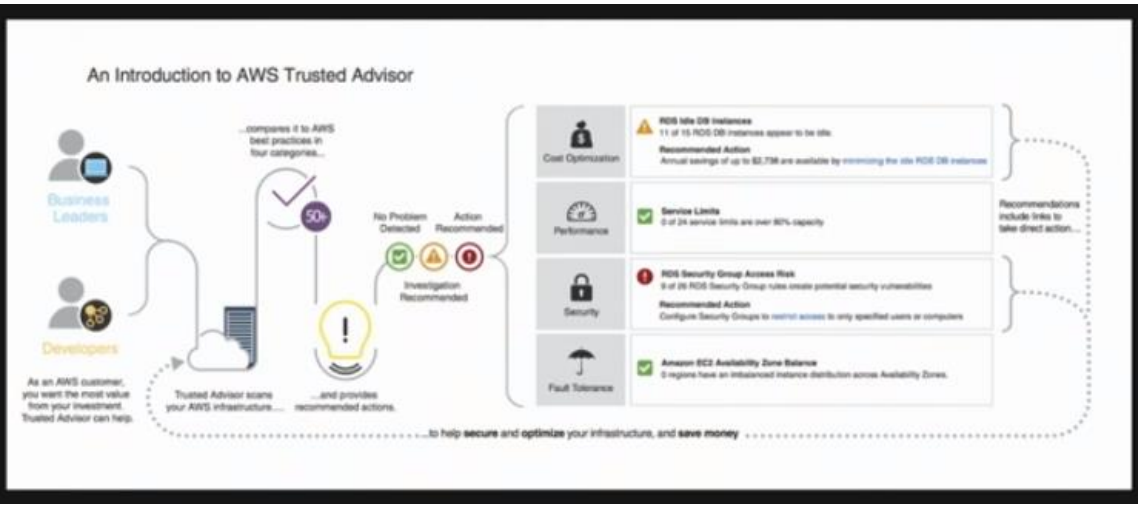
Establishing Best Practices

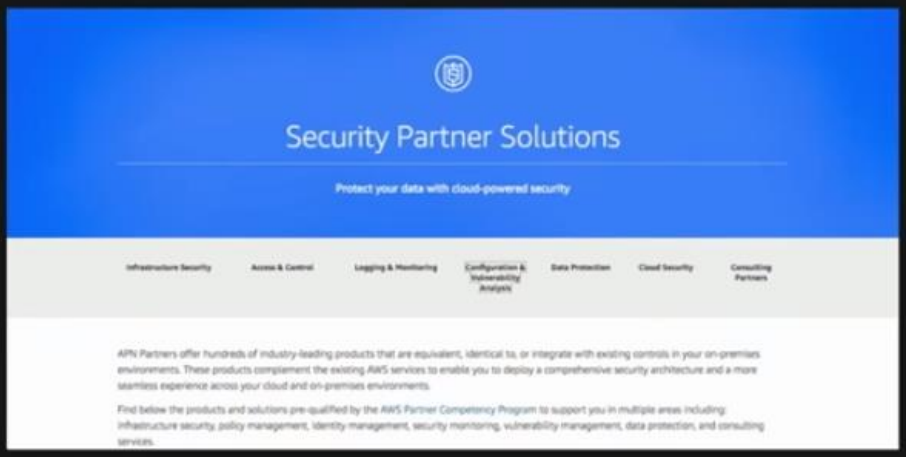
It's a journey...

1. Identify Best Practices
   Learn from mistakes, and, ideally, the mistakes of others
   Use FAQs, troubleshooting guides, and Backup and Recovery steps BEFORE deployment

2. Test Your Assumptions
   Schedule Trial Restores and DR Exercises
   War game scenarios

3. Reassess Frequently
   Follow blogs or the What's New page for new features and announcements
   Schedule periodic architecture reviews with AWS Solutions Architects





https://aws.amazon.com/security/partner-solutions/

https://aws.amazon.com/architecture/well-architected/