GPSTEC319

AWS re:INVENT

Build Once, Deploy Many:

Architecting and Building Automated, Reusable Reference Deployments with AWS CloudFormation

Santiago Cardenas
Partner Solutions Architect
AWS Quick Start

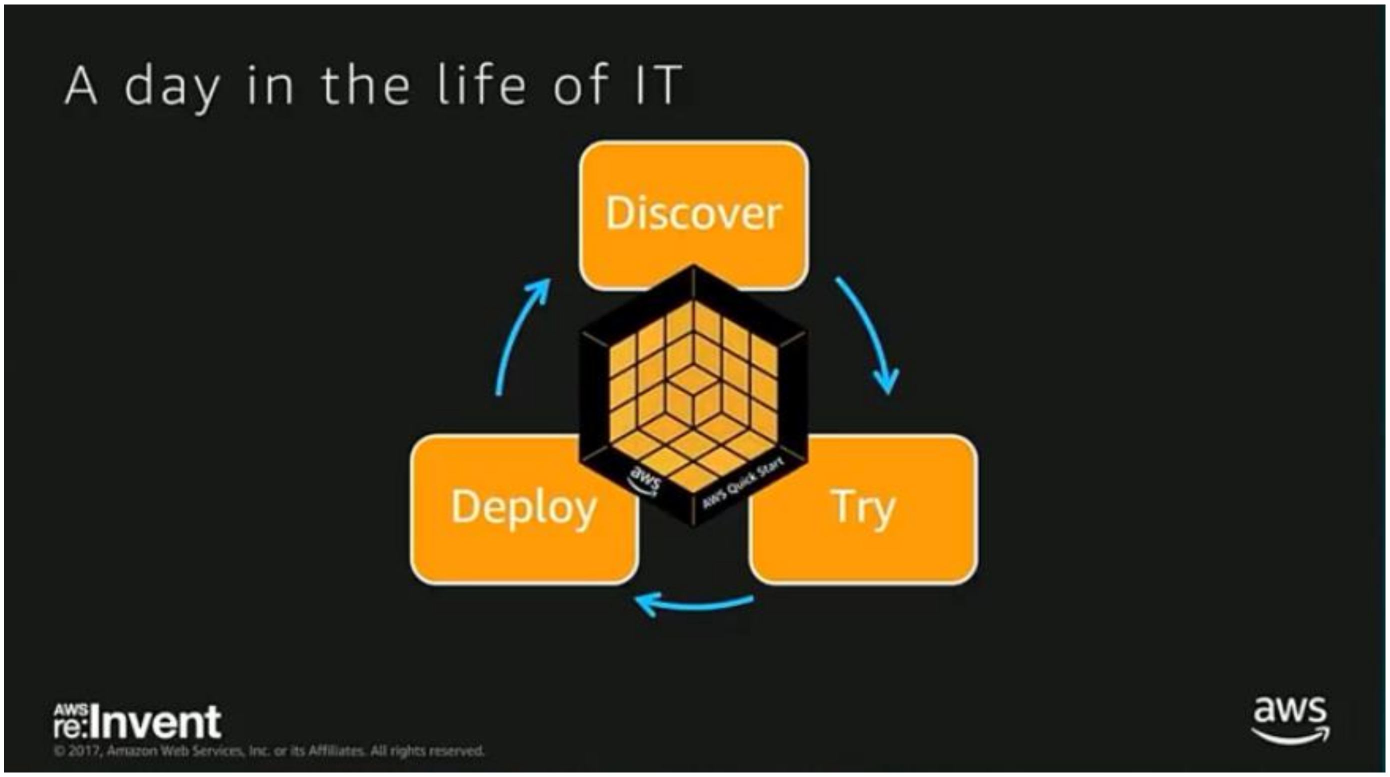
November 28, 2017

This session explains how to build reusable, maintainable AWS CloudFormation–based automation for AWS Cloud deployments. We have built over 50 Quick Start reference deployments with partners and customers, and will share this expertise with you. We explore the anatomy of a typical AWS CloudFormation template, dive deep into best practices for building Quick Start automation across Linux and Windows and explore useful design patterns. This expert-level session is for partners interested in building Quick Starts or other AWS CloudFormation–based automation. It requires familiarity with Git, shell scripting, Windows PowerShell, and AWS services like Amazon EC2, Amazon S3 and AWS CloudFormation.

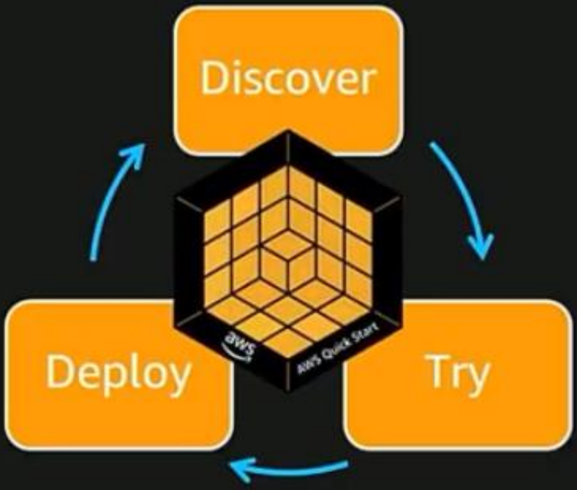
Work smarter, not harder

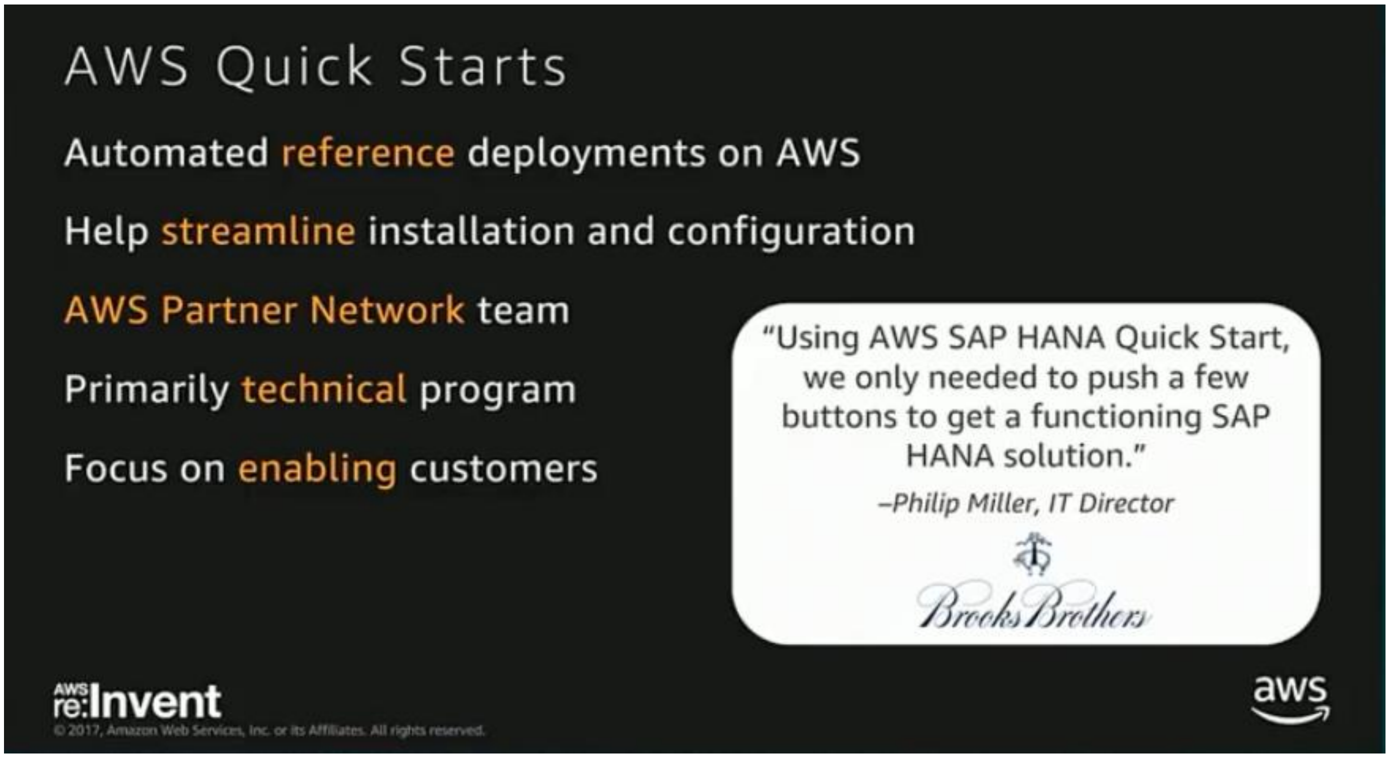
DATA SECURITY
IN THIS CORNER, WE HAVE FIREWALLS, ENCRYPTION, ANTIVIRUS SOFTWARE, ETC. AND IN THIS CORNER, WE HAVE DAVE!!
HUMAN ERROR
www.jklossner.com

copyright 2006 john klossner www.jklossner.com



A day in the life of IT





AWS Quick Starts

Automated reference deployments on AWS

Help streamline installation and configuration

AWS Partner Network team

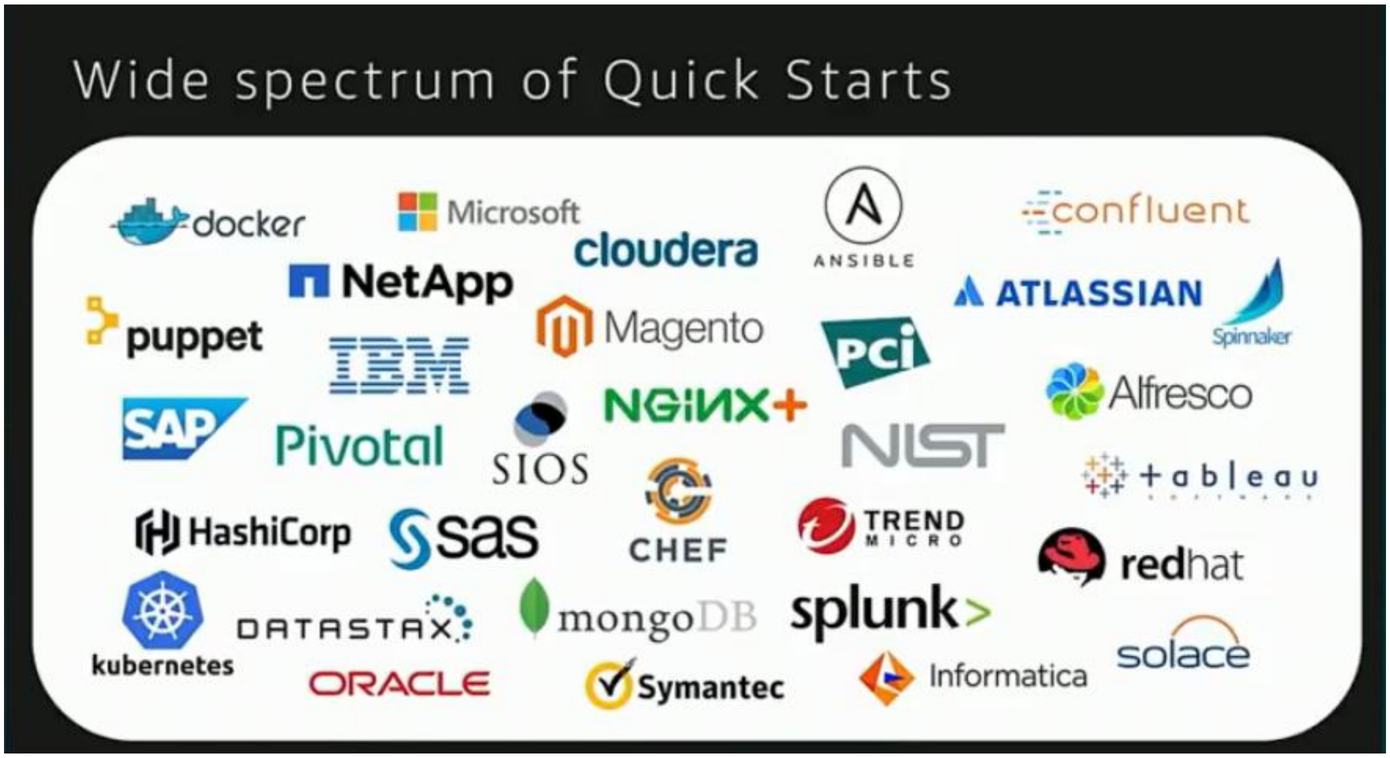
Primarily technical program

Focus on enabling customers

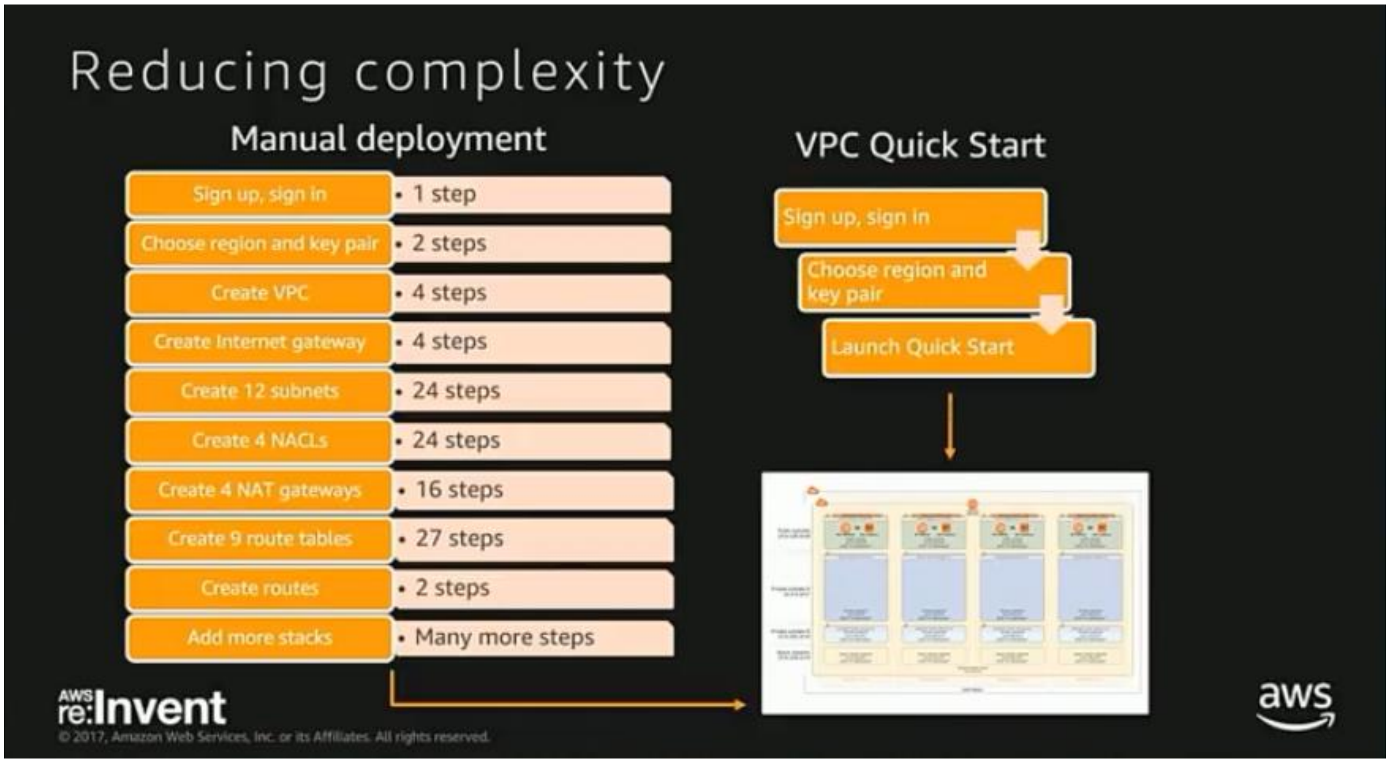
"Using AWS SAP HANA Quick Start, we only needed to push a few buttons to get a functioning SAP HANA solution."

–Philip Miller, IT Director

Brooks Brothers



Wide spectrum of Quick Starts



Reducing complexity

Manual deployment

Sign up, sign in	• 1 step
Choose region and key pair	• 2 steps
Create VPC	• 4 steps
Create Internet gateway	• 4 steps
Create 12 subnets	• 24 steps
Create 4 NACLs	• 24 steps
Create 4 NAT gateways	• 16 steps
Create 9 route tables	• 27 steps
Create routes	• 2 steps
Add more stacks	• Many more steps

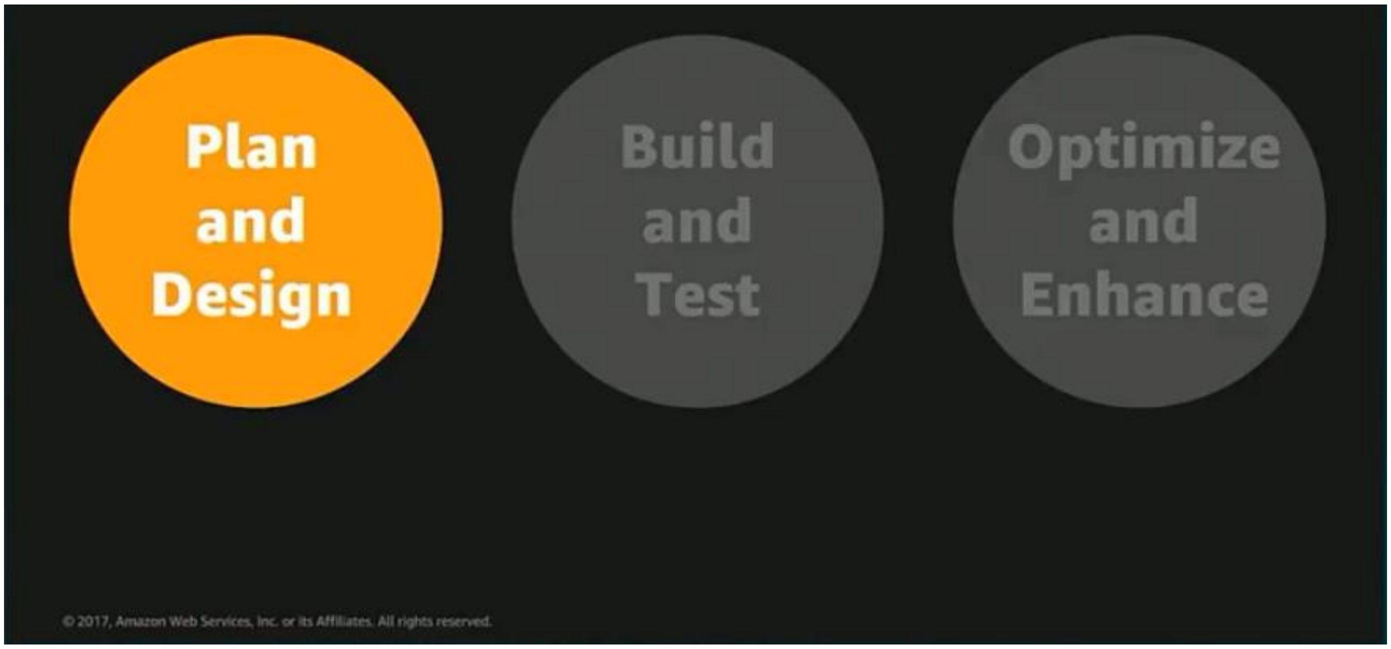
VPC Quick Start

Sign up, sign in

Choose region and key pair

Launch Quick Start

The above chart is for building a VPC that is multi-AZ, different CIDR blocks, private and public subnets, etc.

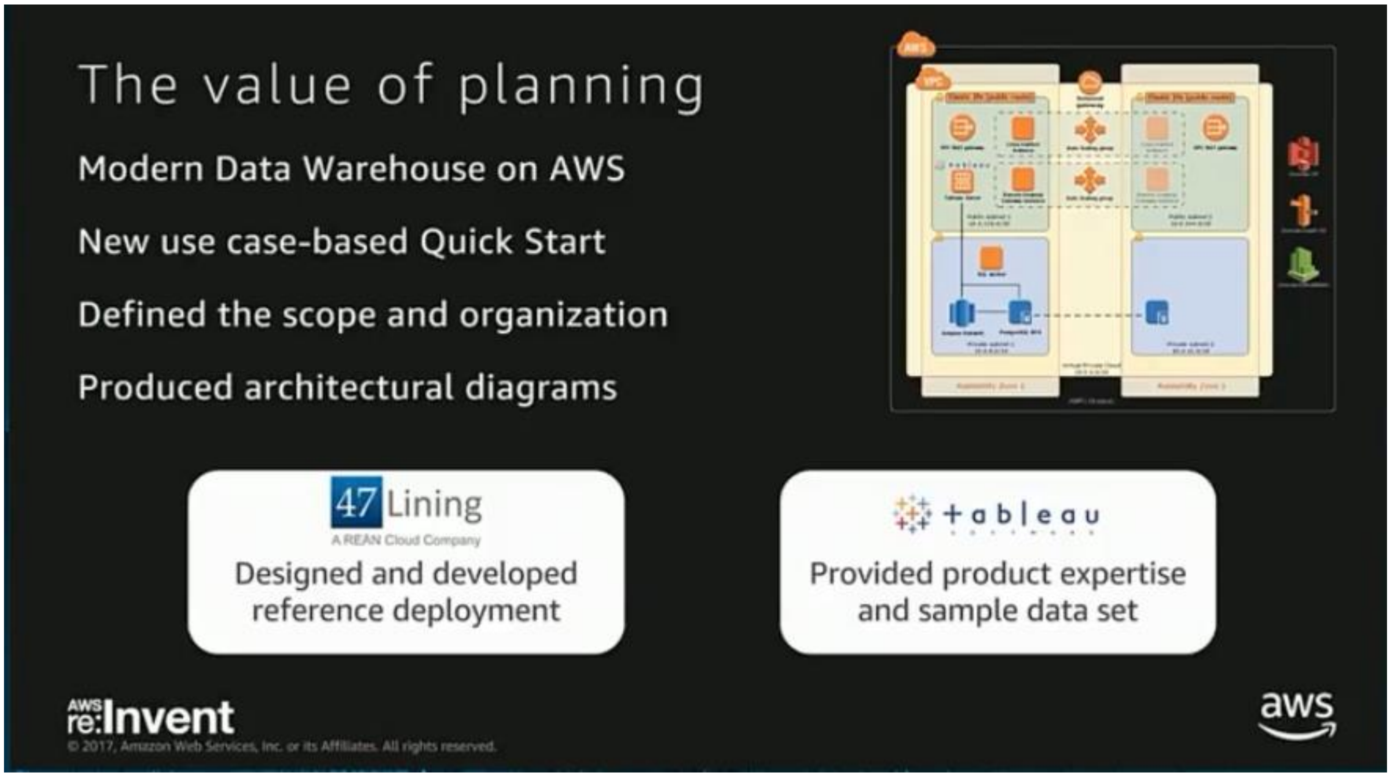


Plan and Design

Build and Test

Optimize and Enhance

We will now look at the different stages of building and deploying and reference quick start architecture



The value of planning

Modern Data Warehouse on AWS

New use case-based Quick Start

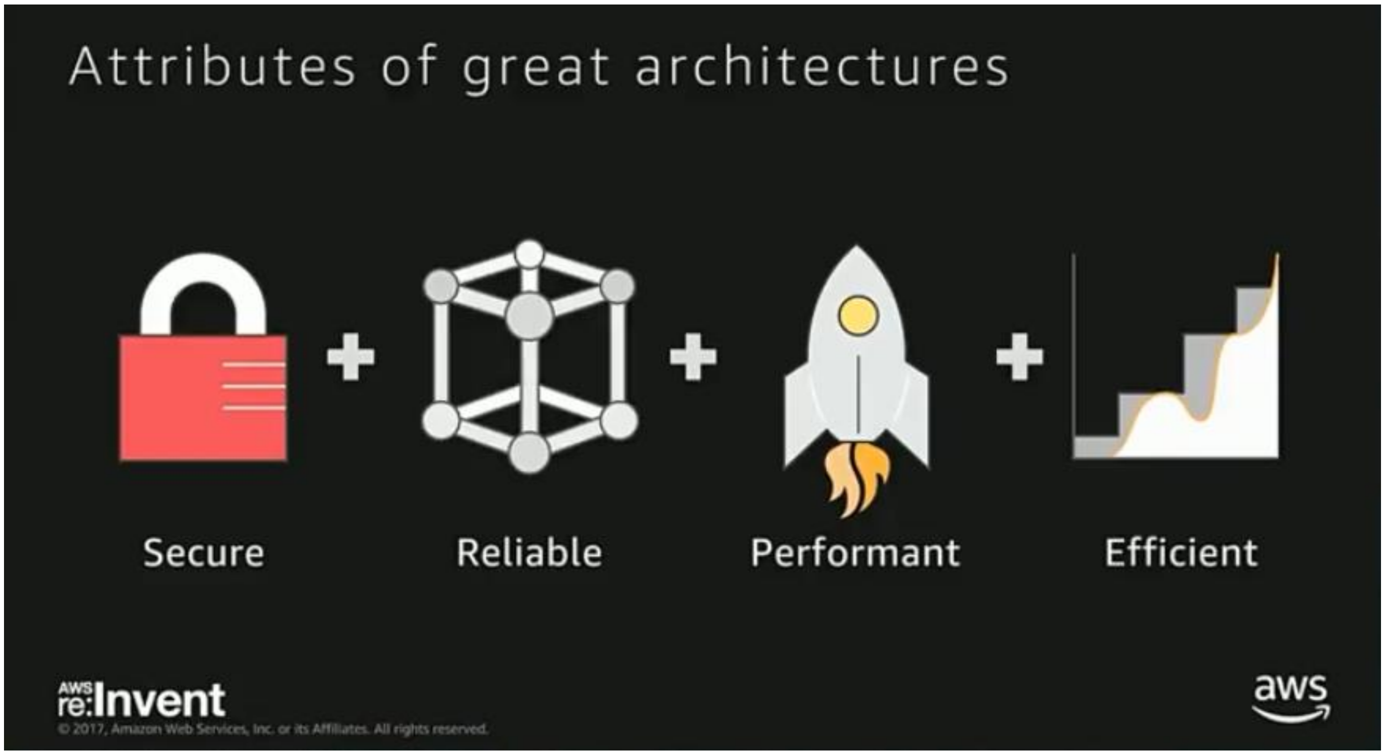
Defined the scope and organization

Produced architectural diagrams

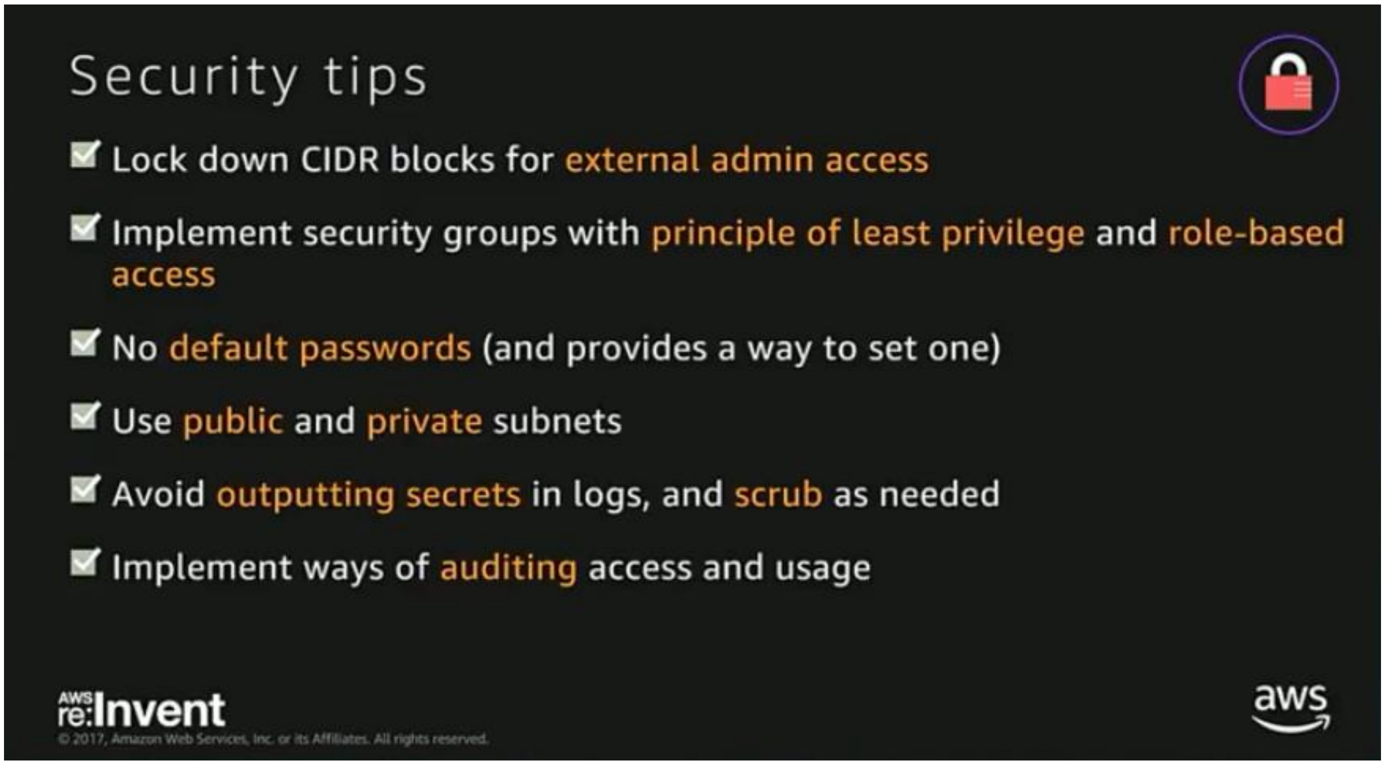
47 Lining — A REAN Cloud Company: Designed and developed reference deployment

tableau: Provided product expertise and sample data set

We had to define the scope of the work and the organization of the assets, this allowed us to build some very clear architectural diagrams that shows the instances, subnets, SGs, services, etc.



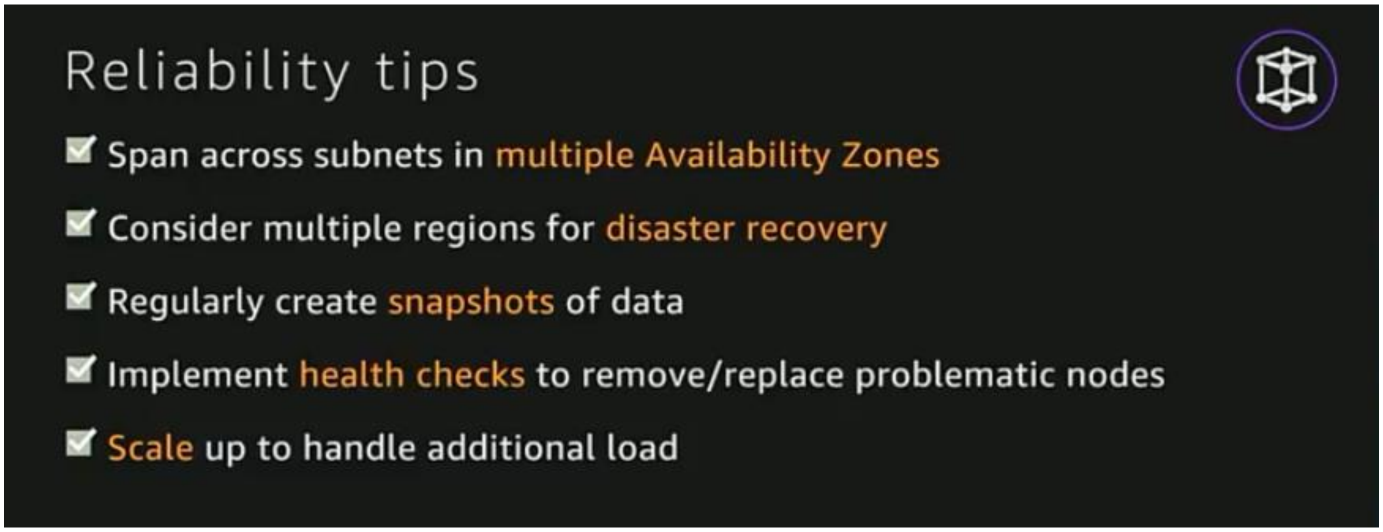
Attributes of great architectures



Security tips

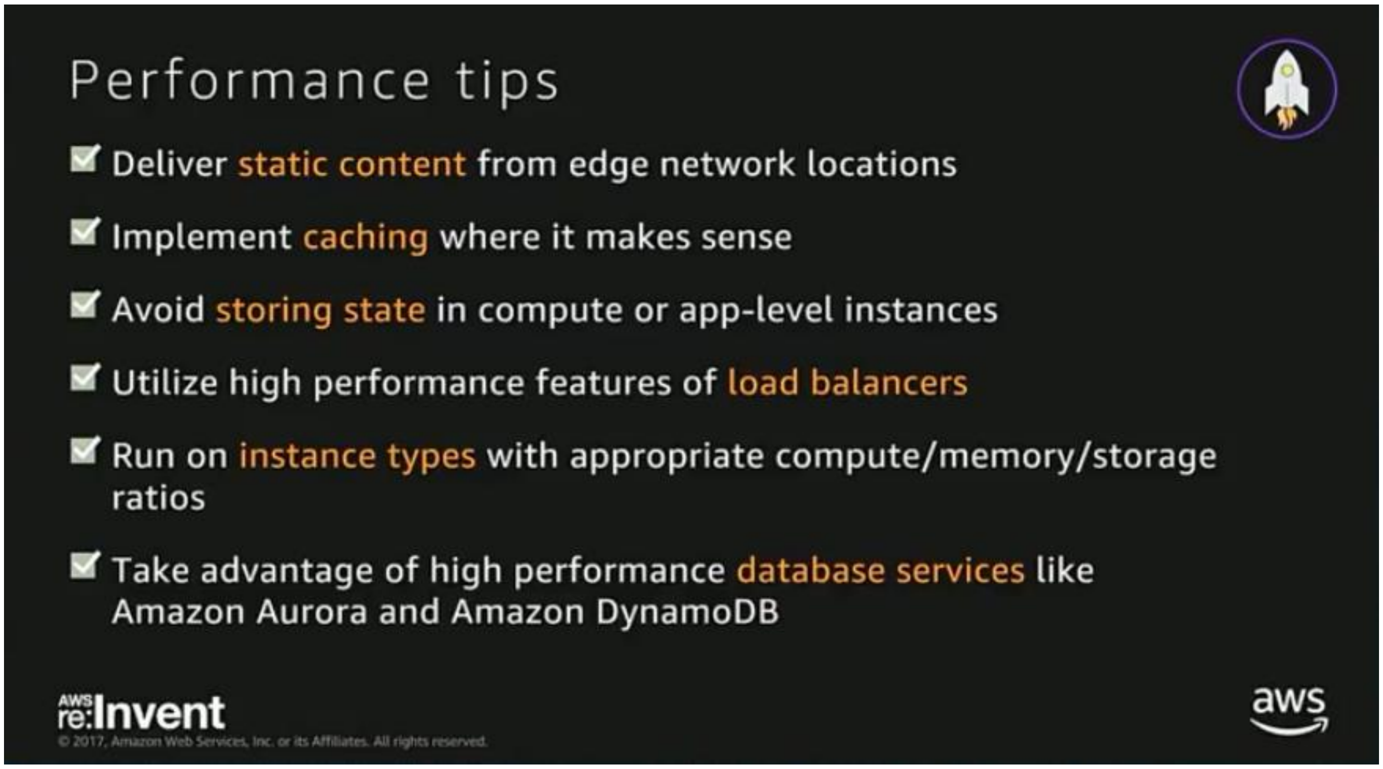
- ✓ Lock down CIDR blocks for external admin access
- ✓ Implement security groups with principle of least privilege and role-based access
- ✓ No default passwords (and provides a way to set one)
- ✓ Use public and private subnets
- ✓ Avoid outputting secrets in logs, and scrub as needed
- ✓ Implement ways of auditing access and usage

Always provide lock down CIDR blocks for external admin access like SSH, RDP, etc.



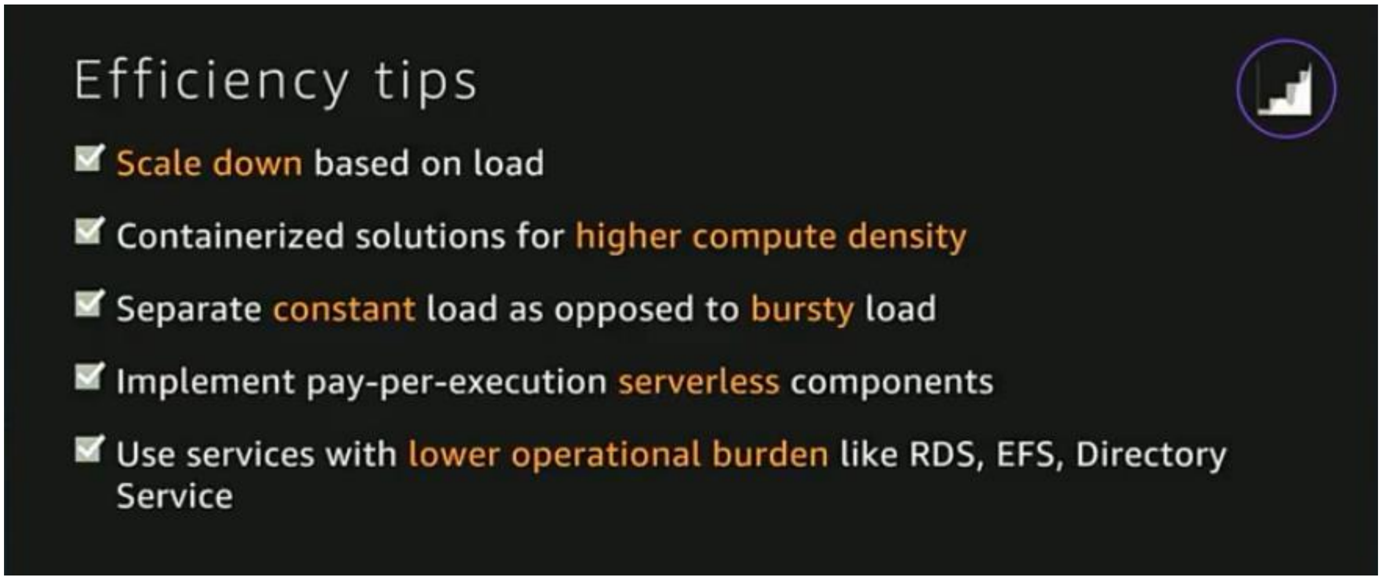
Reliability tips

- ✓ Span across subnets in multiple Availability Zones
- ✓ Consider multiple regions for disaster recovery
- ✓ Regularly create snapshots of data
- ✓ Implement health checks to remove/replace problematic nodes
- ✓ Scale up to handle additional load



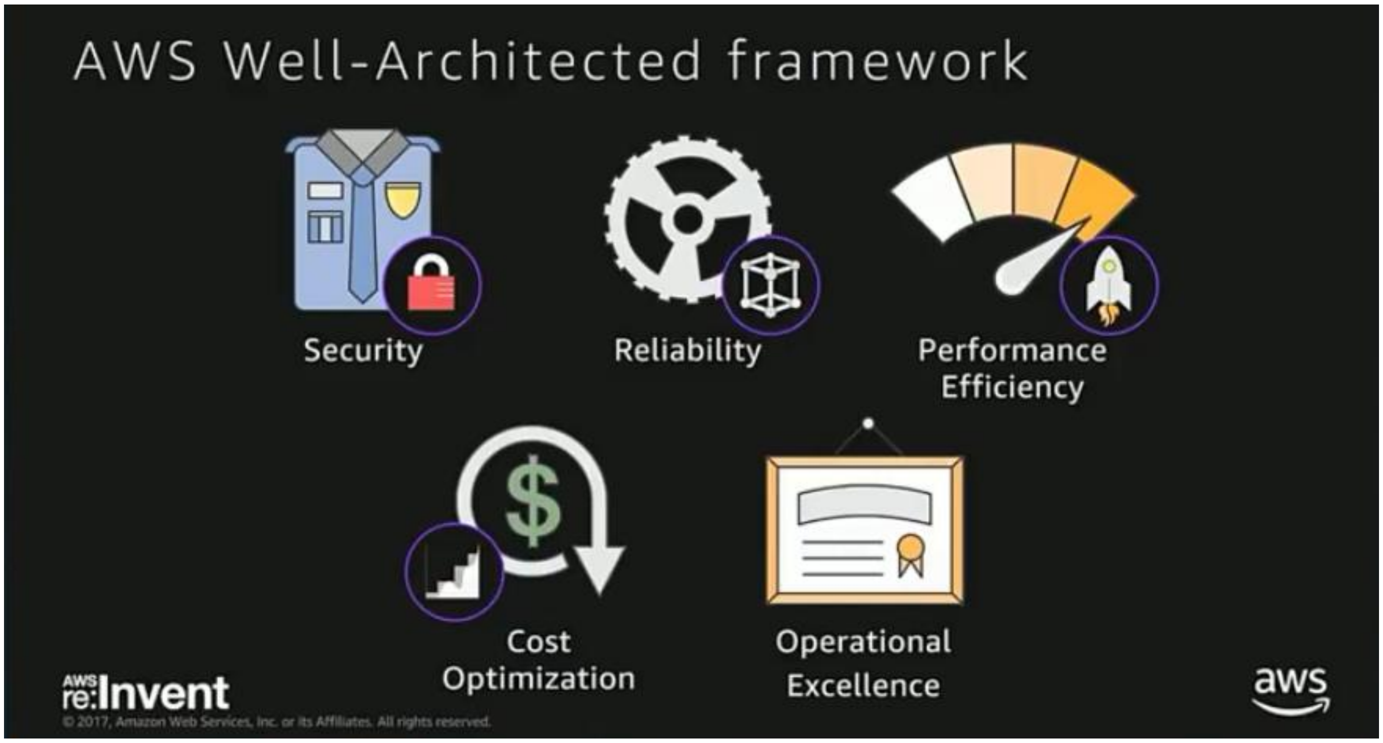
Performance tips

- ✓ Deliver static content from edge network locations
- ✓ Implement caching where it makes sense
- ✓ Avoid storing state in compute or app-level instances
- ✓ Utilize high performance features of load balancers
- ✓ Run on instance types with appropriate compute/memory/storage ratios
- ✓ Take advantage of high performance database services like Amazon Aurora and Amazon DynamoDB



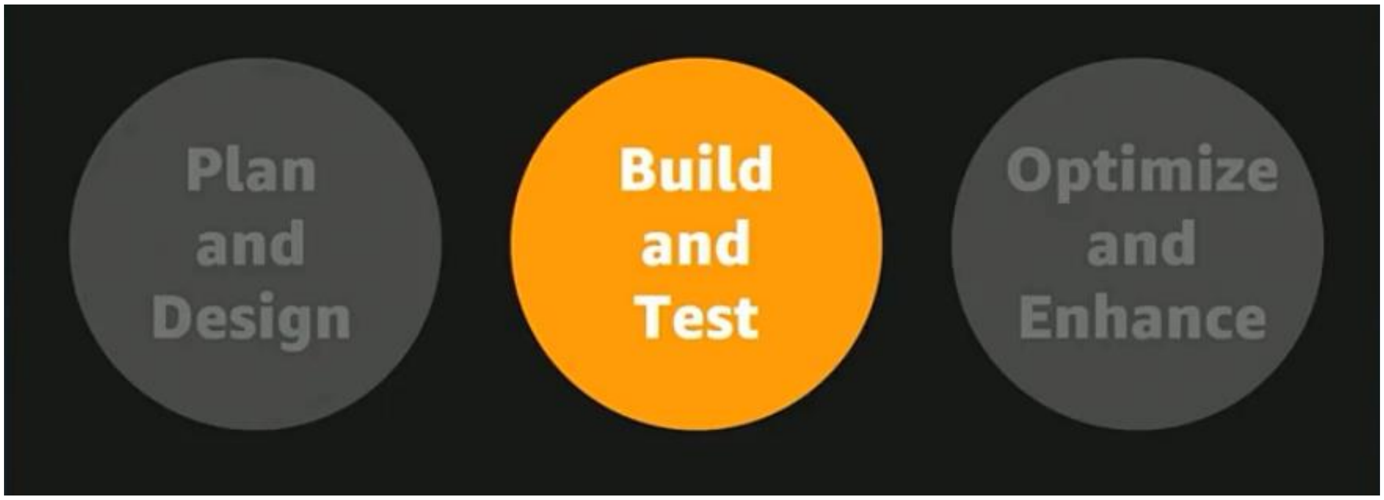
Efficiency tips

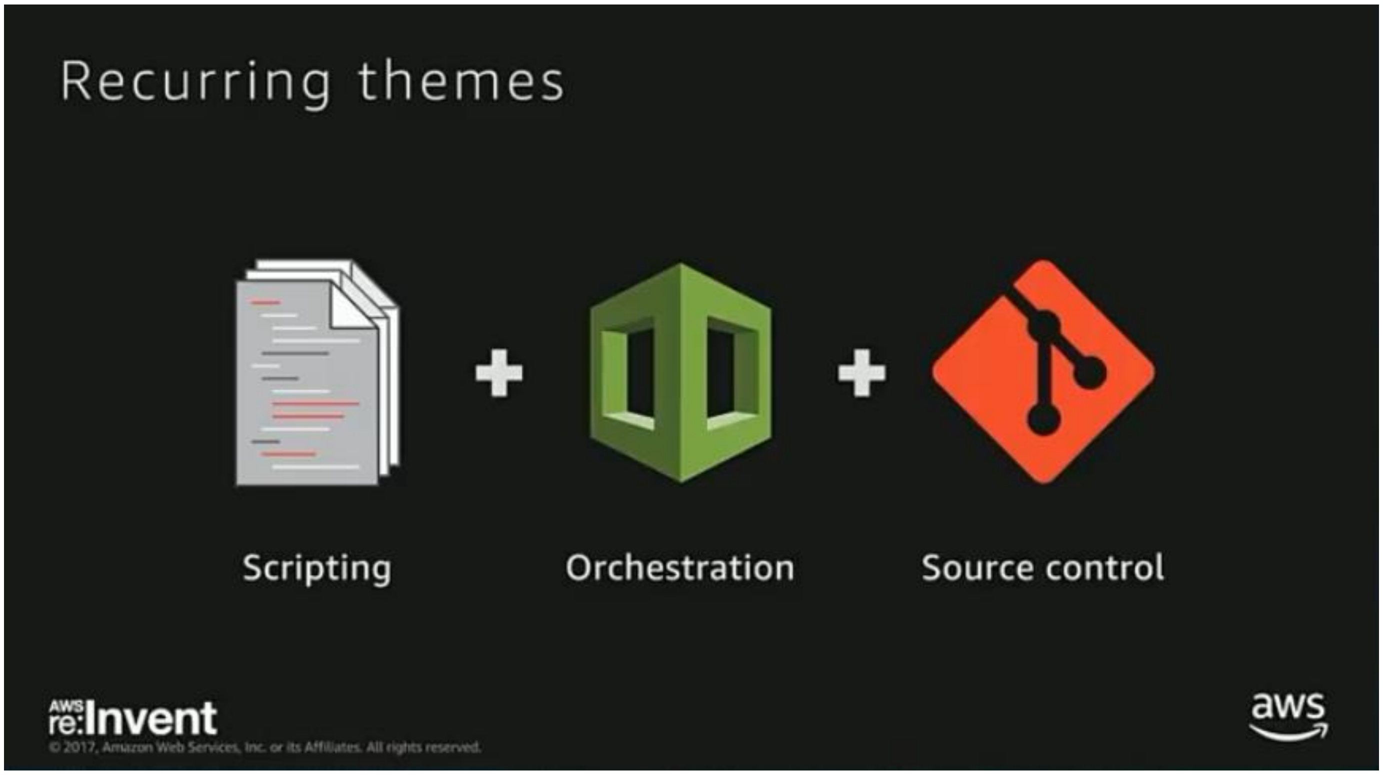
- ✓ Scale down based on load
- ✓ Containerized solutions for higher compute density
- ✓ Separate constant load as opposed to bursty load
- ✓ Implement pay-per-execution serverless components
- ✓ Use services with lower operational burden like RDS, EFS, Directory Service



AWS Well-Architected framework

Security
Reliability
Performance Efficiency
Cost Optimization
Operational Excellence
AWS re:Invent
aws



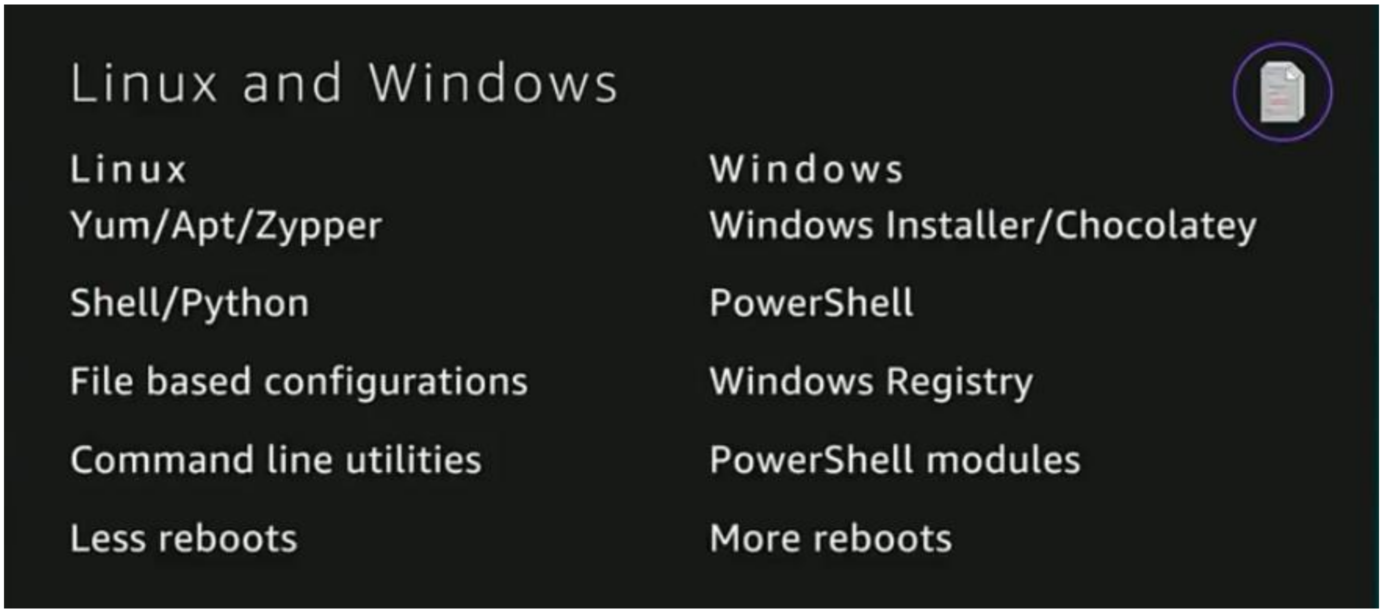


Recurring themes

re:Invent

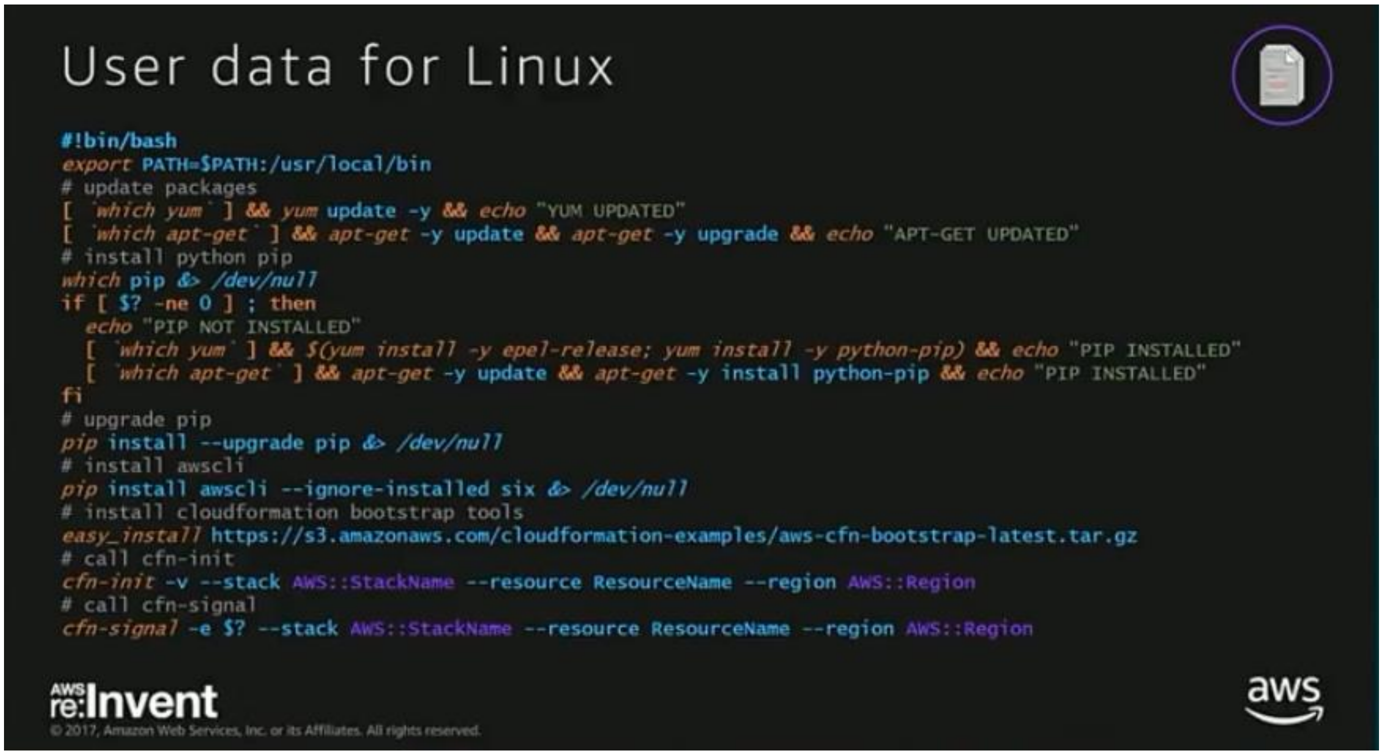
aws

This covers the infrastructure-as-code process for most implementations we have seen



Linux and Windows

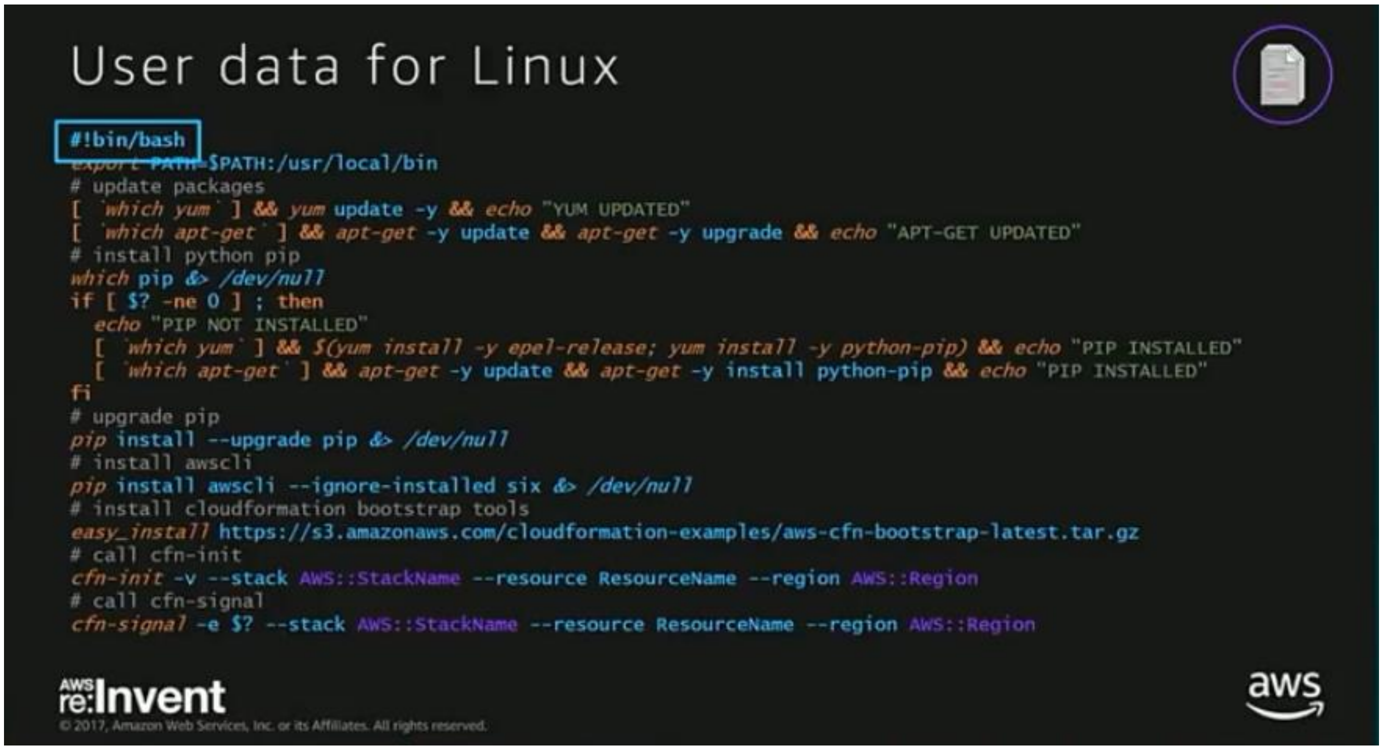
Linux	Windows
Yum/Apt/Zypper	Windows Installer/Chocolatey
Shell/Python	PowerShell
File based configurations	Windows Registry
Command line utilities	PowerShell modules
Less reboots	More reboots



User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

We have user data on instances, this gives us the ability to execute commands when the instances are being launched.

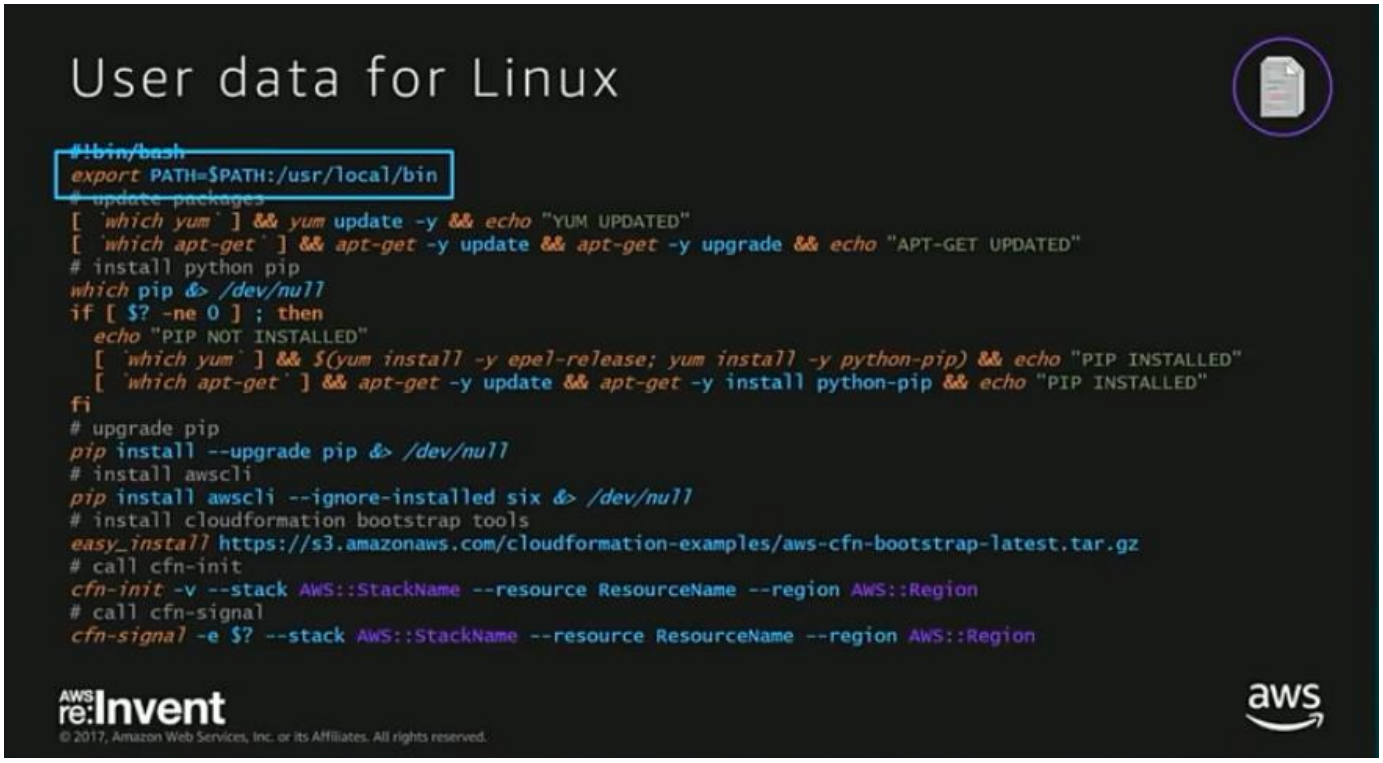


User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

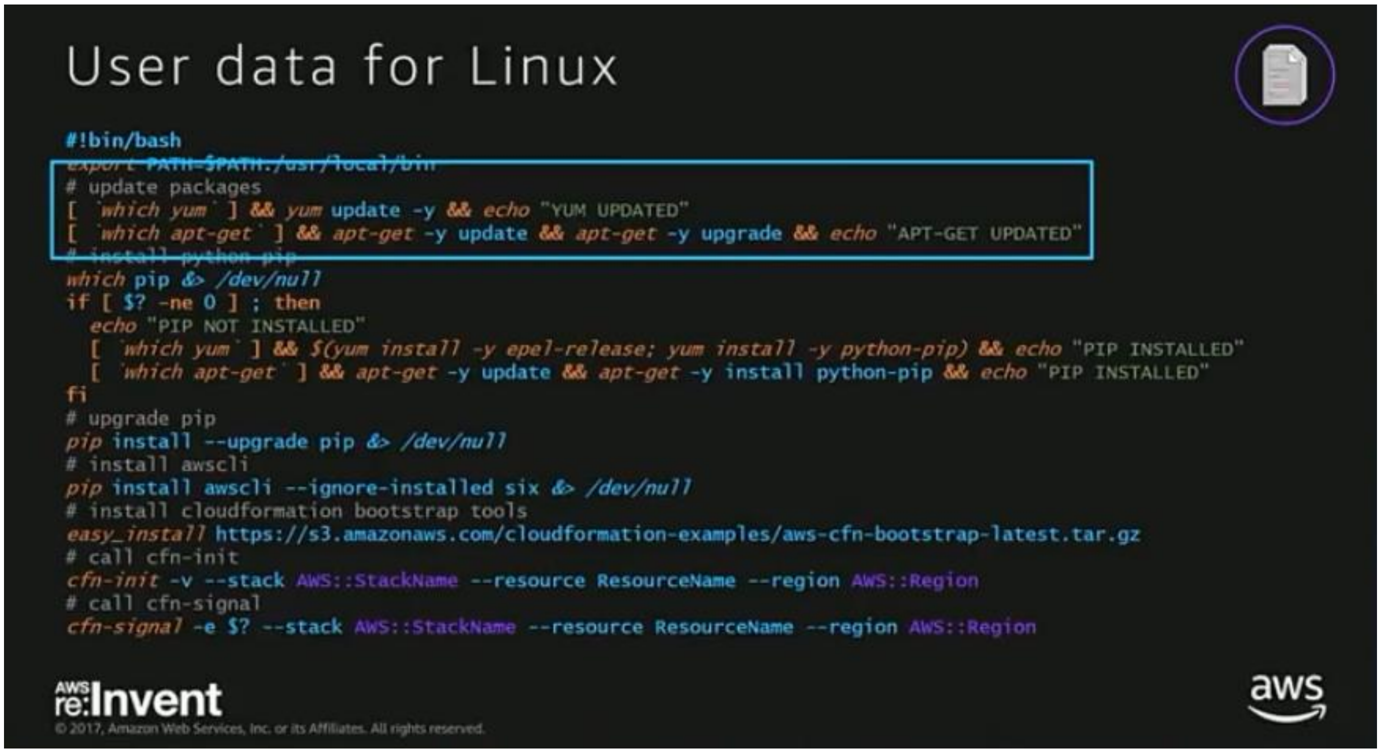
We start by specifying that we are using bash as the interpreter for the script



User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

Then we set up some extra path so that we can account for some of the tools that we will be installing later

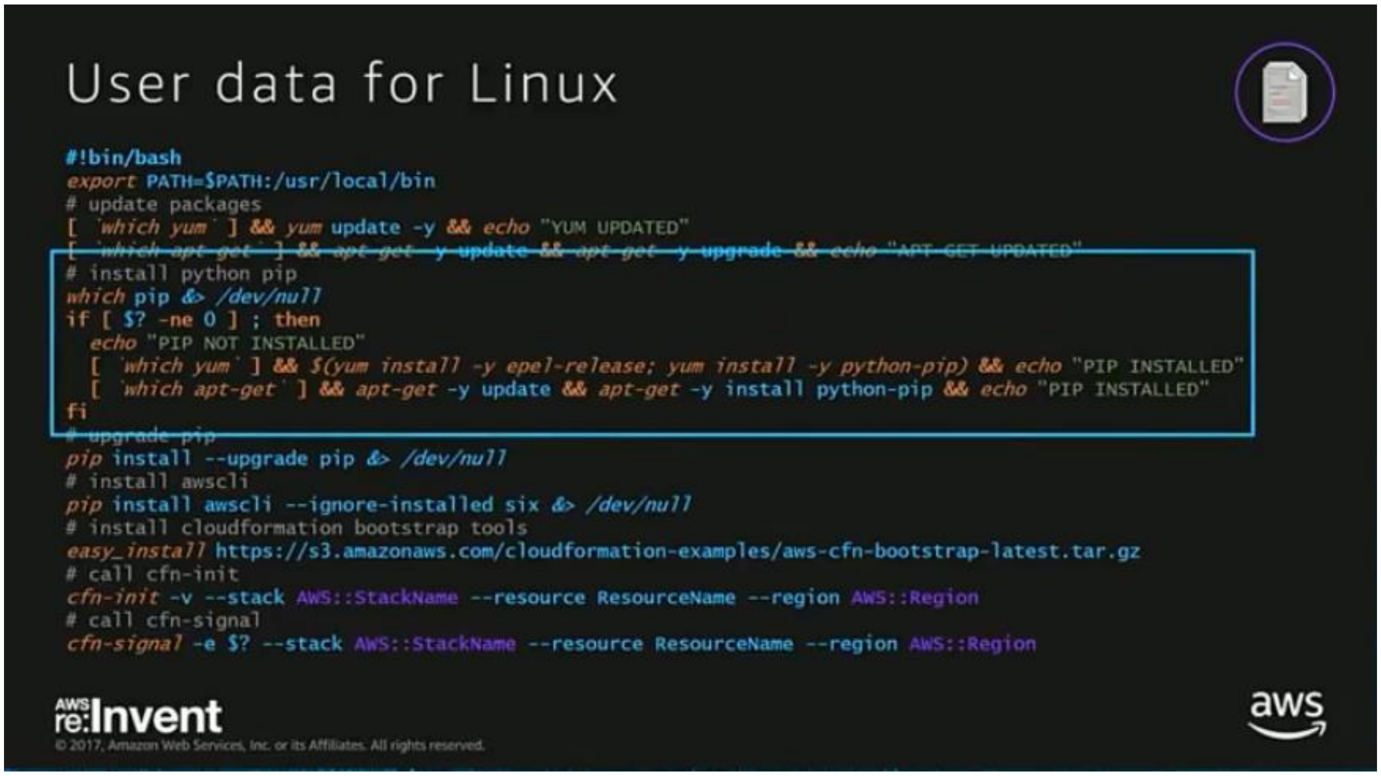


User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

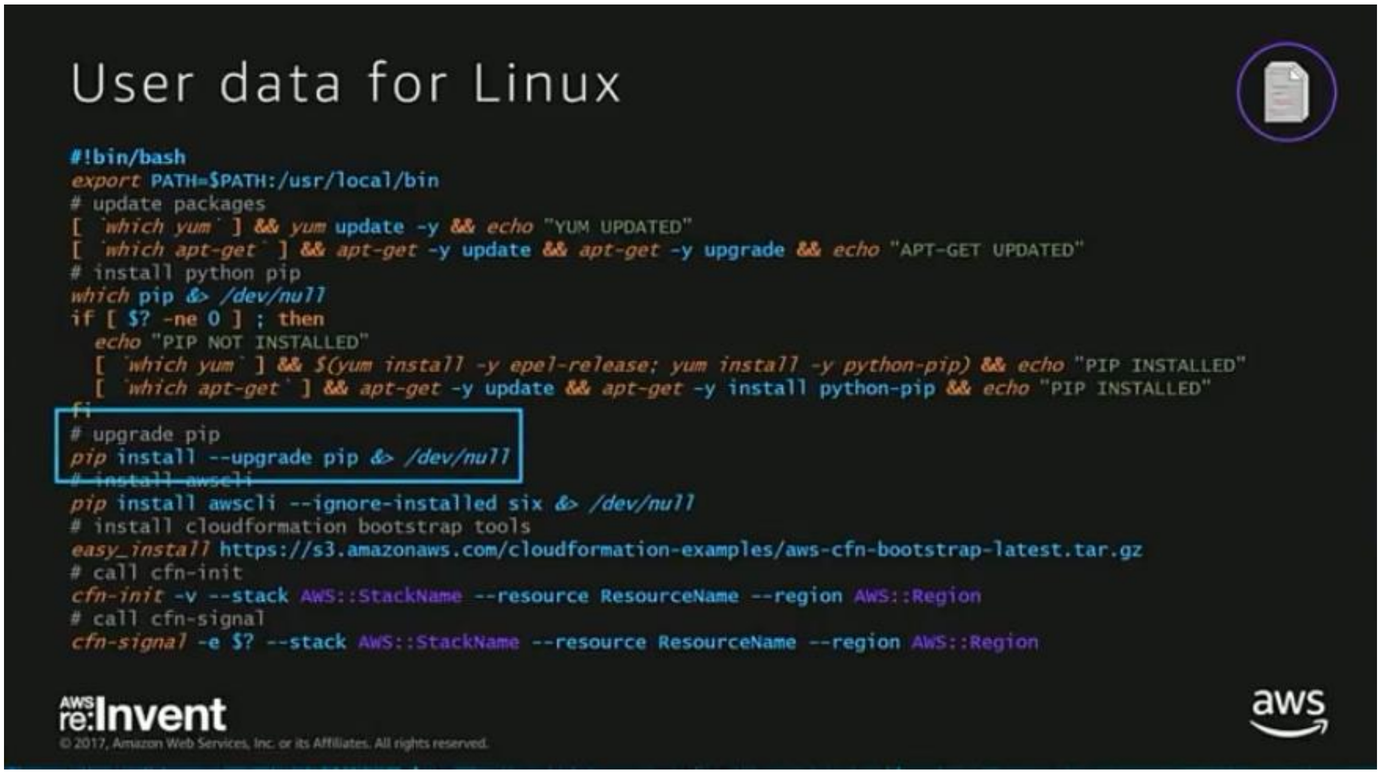
We then update the operating system



User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

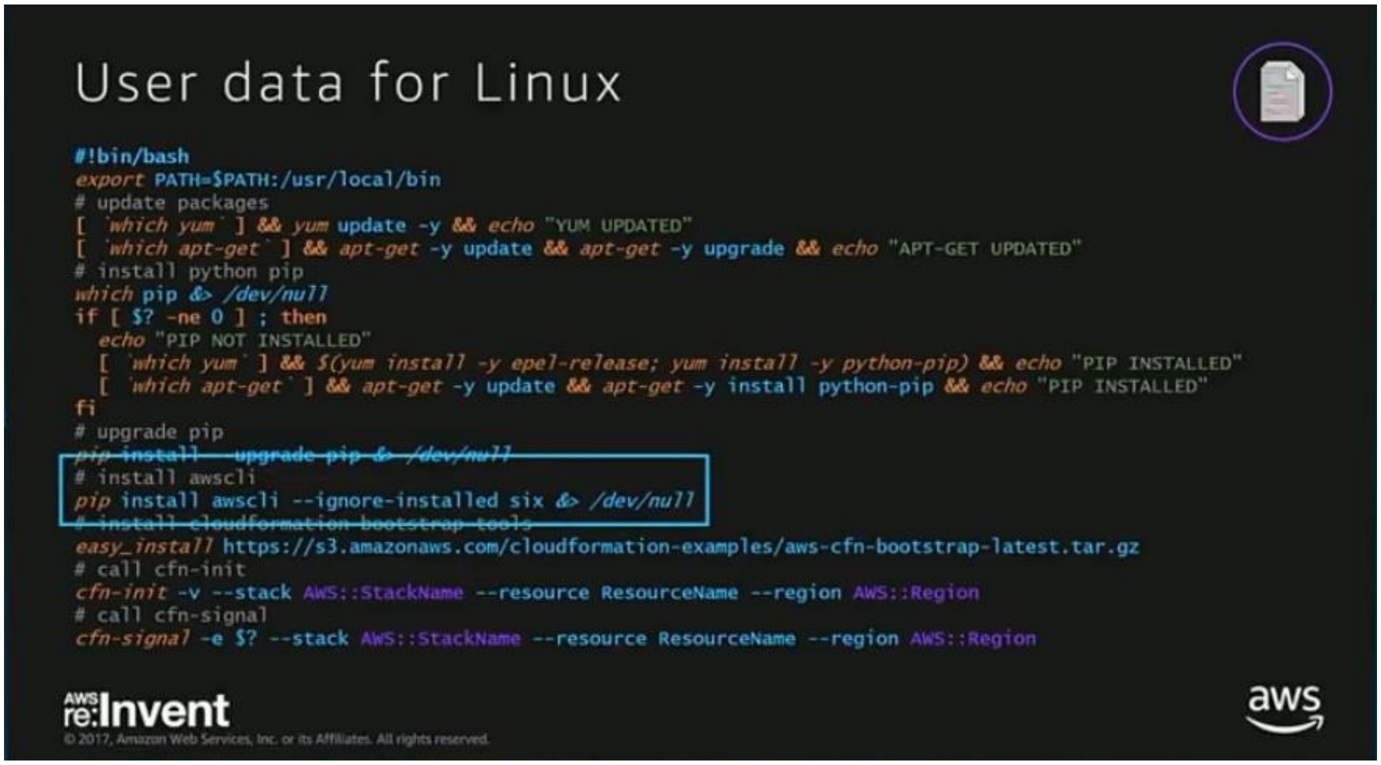
We install python and pip for package management for python



User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

We upgrade pip to the latest version available



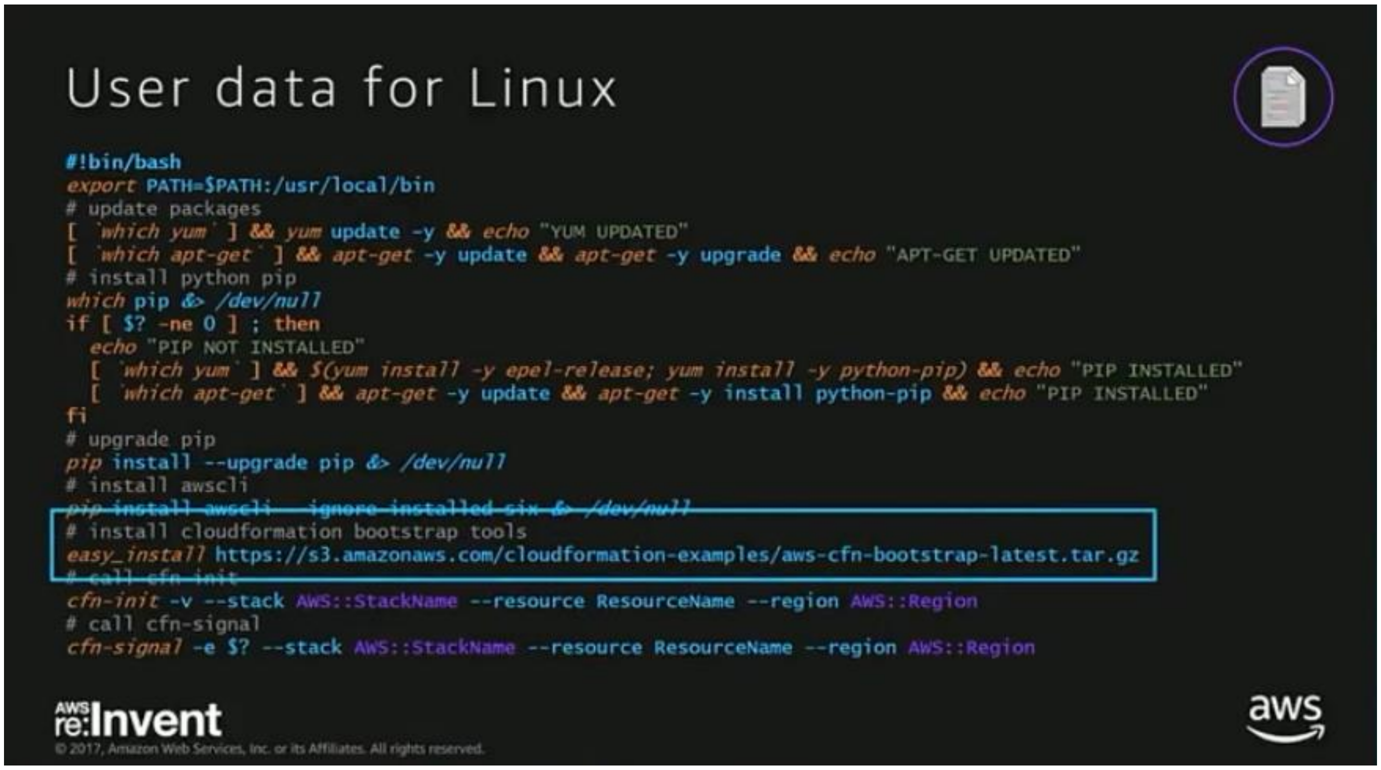
User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

re:Invent © 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

We then install the AWS CLI for sending calls to AWS Service APIs



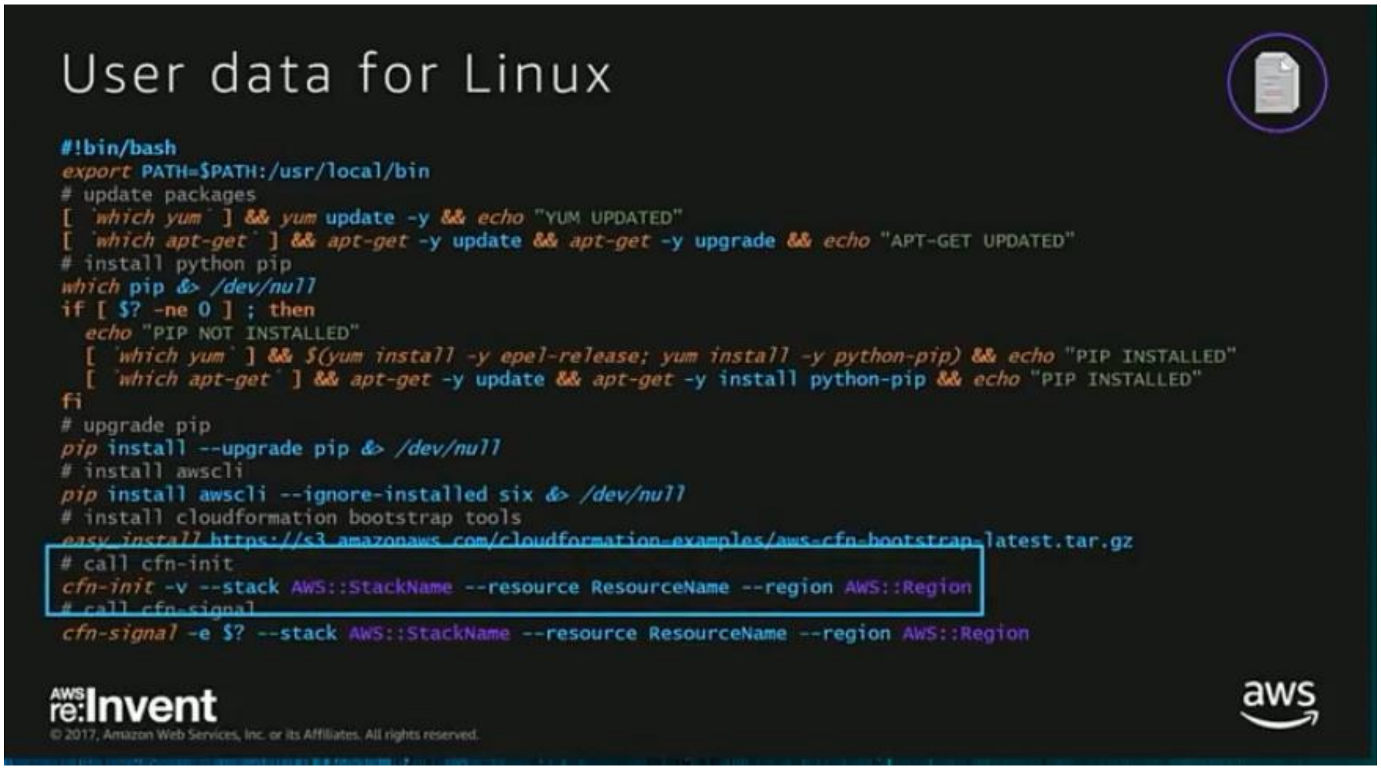
User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

re:Invent © 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

aws

Then we install the CloudFormation bootstrap tools tat will give use tools like cfn-init



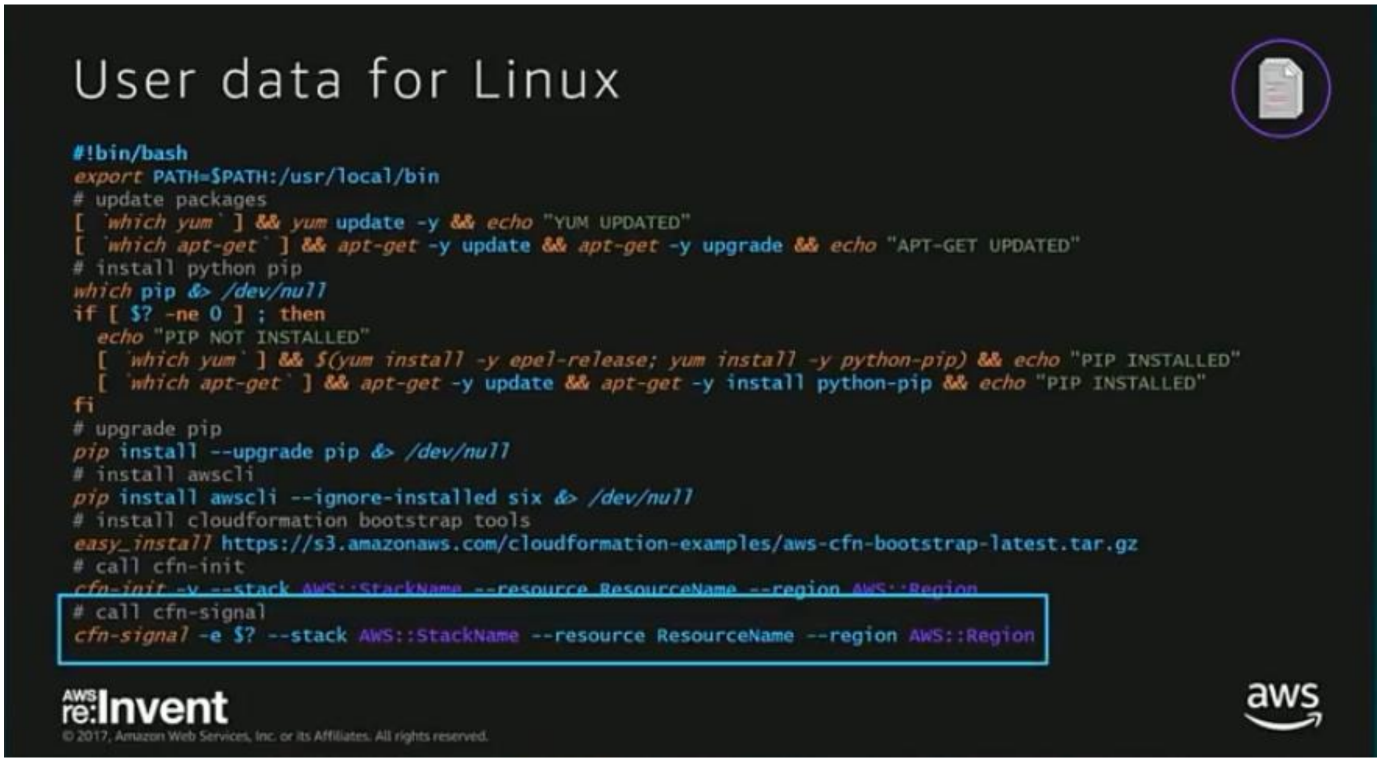
User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

We then use cfn-init for configuring our infrastructure



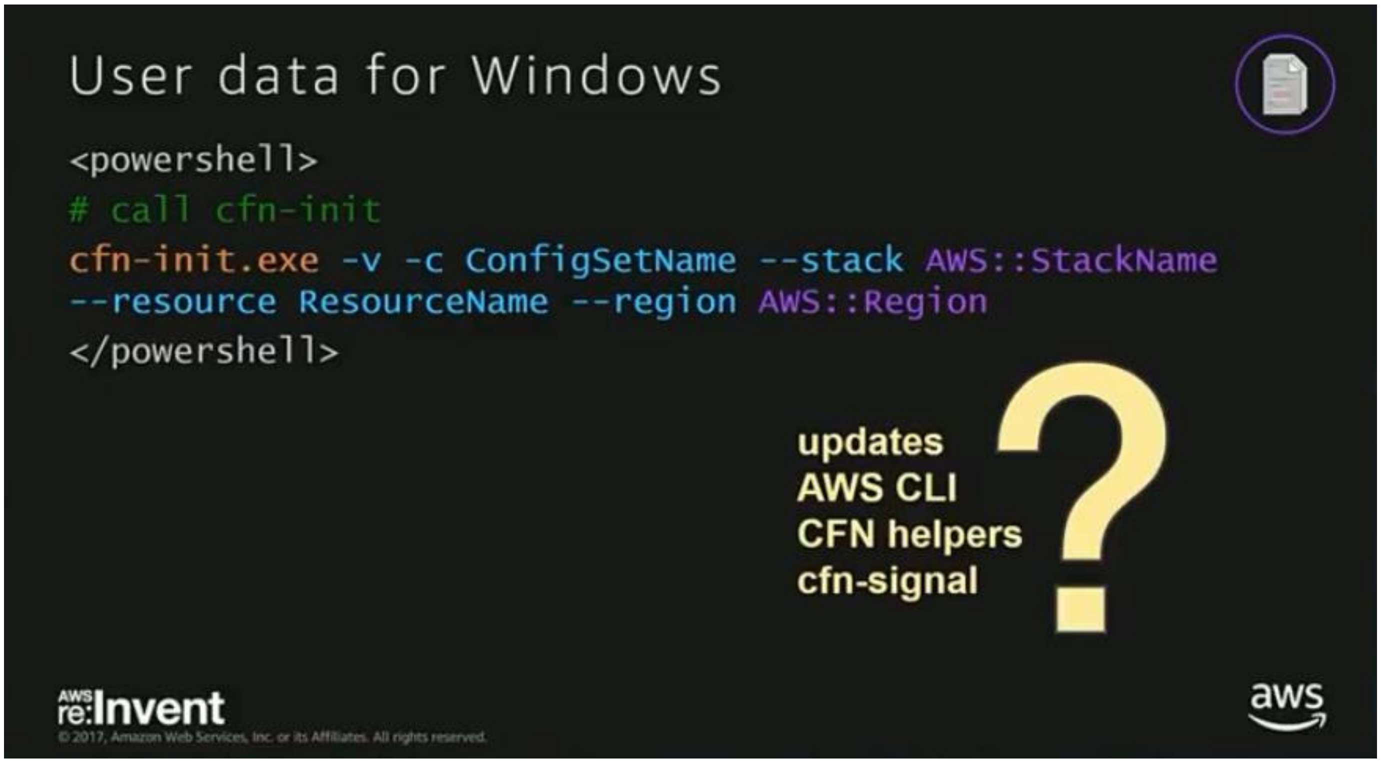
User data for Linux

```
#!bin/bash
export PATH=$PATH:/usr/local/bin
# update packages
[ `which yum` ] && yum update -y && echo "YUM UPDATED"
[ `which apt-get` ] && apt-get -y update && apt-get -y upgrade && echo "APT-GET UPDATED"
# install python pip
which pip &> /dev/null
if [ $? -ne 0 ] ; then
  echo "PIP NOT INSTALLED"
  [ `which yum` ] && $(yum install -y epel-release; yum install -y python-pip) && echo "PIP INSTALLED"
  [ `which apt-get` ] && apt-get -y update && apt-get -y install python-pip && echo "PIP INSTALLED"
fi
# upgrade pip
pip install --upgrade pip &> /dev/null
# install awscli
pip install awscli --ignore-installed six &> /dev/null
# install cloudformation bootstrap tools
easy_install https://s3.amazonaws.com/cloudformation-examples/aws-cfn-bootstrap-latest.tar.gz
# call cfn-init
cfn-init -v --stack AWS::StackName --resource ResourceName --region AWS::Region
# call cfn-signal
cfn-signal -e $? --stack AWS::StackName --resource ResourceName --region AWS::Region
```

AWS re:Invent

© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

We then finalize the script with cfn-signal to signal that configuration is completed.

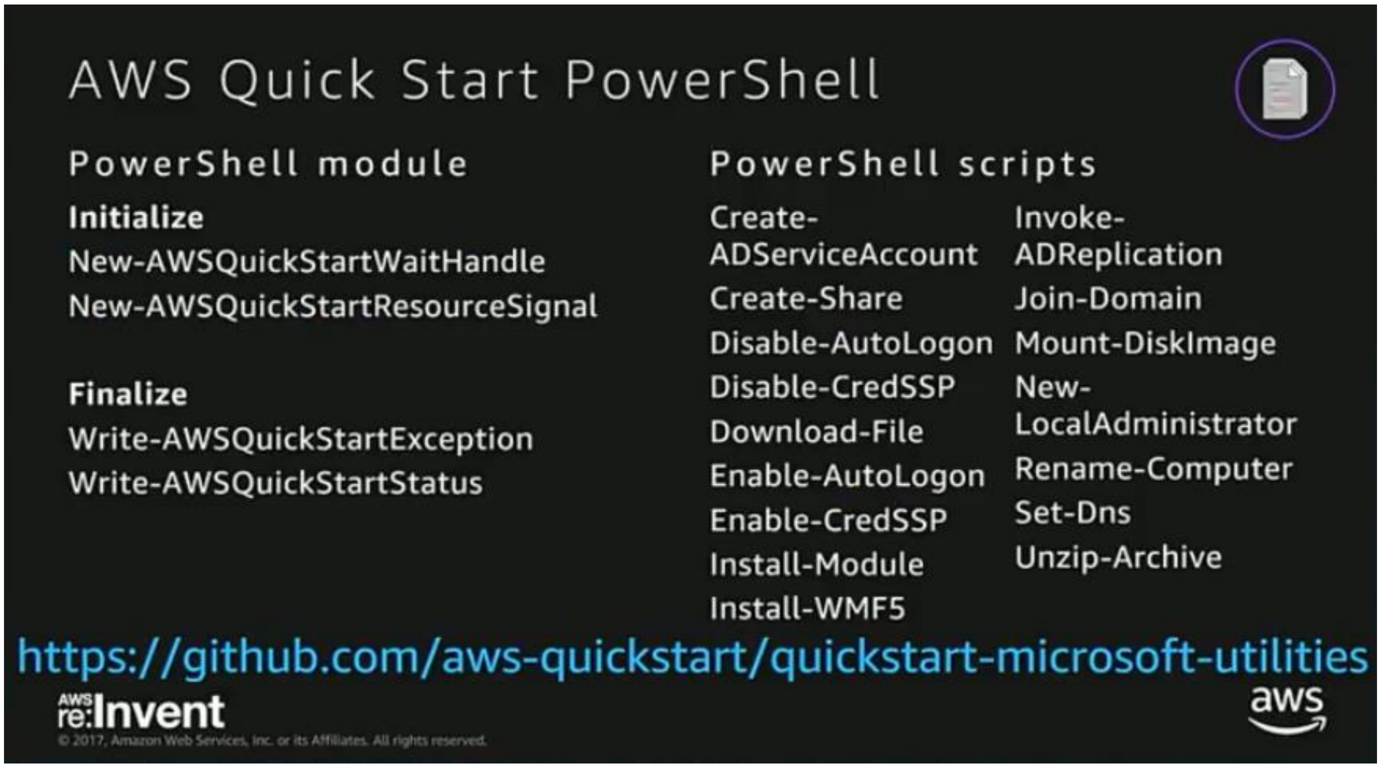


User data for Windows

```
<powershell>
# call cfn-init
cfn-init.exe -v -c ConfigSetName --stack AWS::StackName
--resource ResourceName --region AWS::Region
</powershell>
```

updates
AWS CLI
CFN helpers
cfn-signal
?

Windows always have fresh AMIs that are less that 2 months old with the cfn helpers pre-installed.



AWS Quick Start PowerShell

PowerShell module

Initialize

New-AWSQuickStartWaitHandle

New-AWSQuickStartResourceSignal

Finalize

Write-AWSQuickStartException

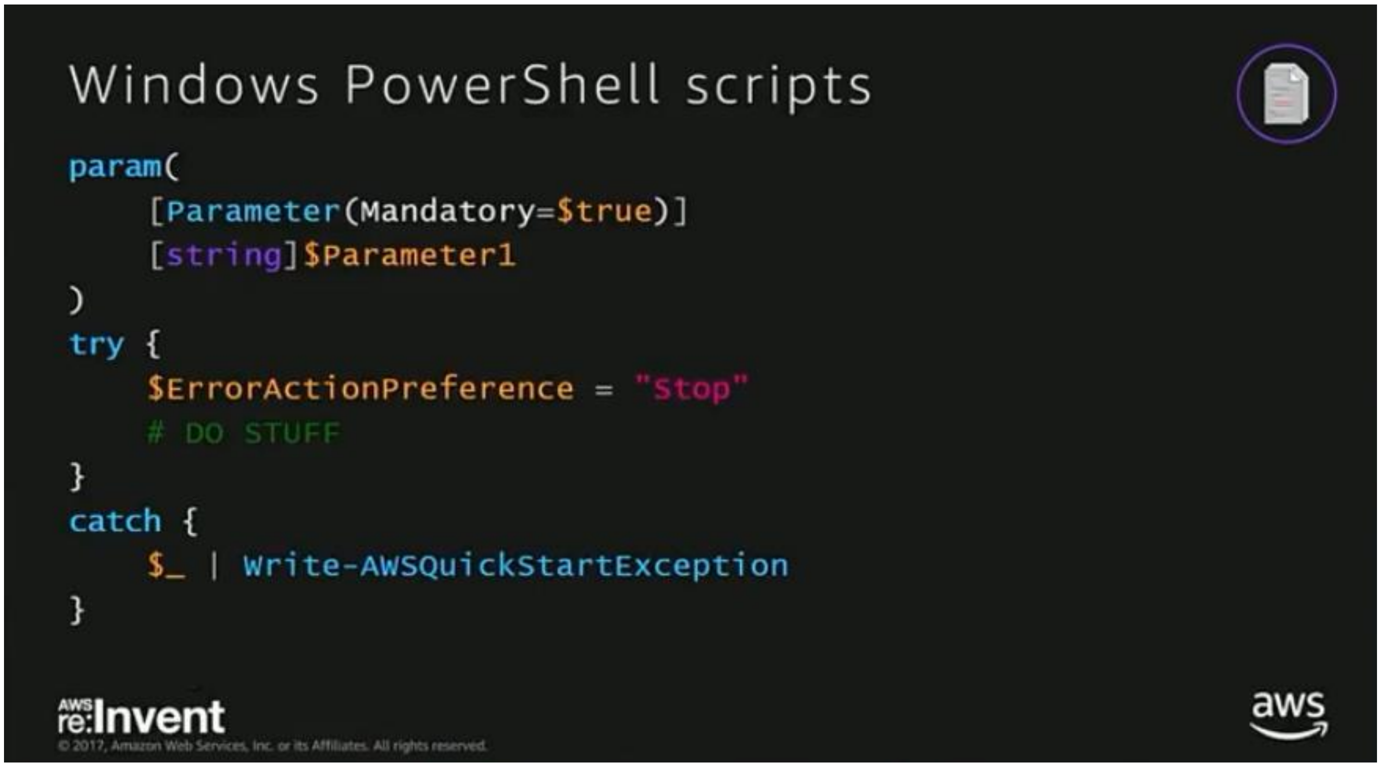
Write-AWSQuickStartStatus

PowerShell scripts

- Create-ADServiceAccount
- Create-Share
- Disable-AutoLogon
- Disable-CredSSP
- Download-File
- Enable-AutoLogon
- Enable-CredSSP
- Install-Module
- Install-WMF5
- Invoke-ADReplication
- Join-Domain
- Mount-DiskImage
- New-LocalAdministrator
- Rename-Computer
- Set-Dns
- Unzip-Archive

https://github.com/aws-quickstart/quickstart-microsoft-utilities

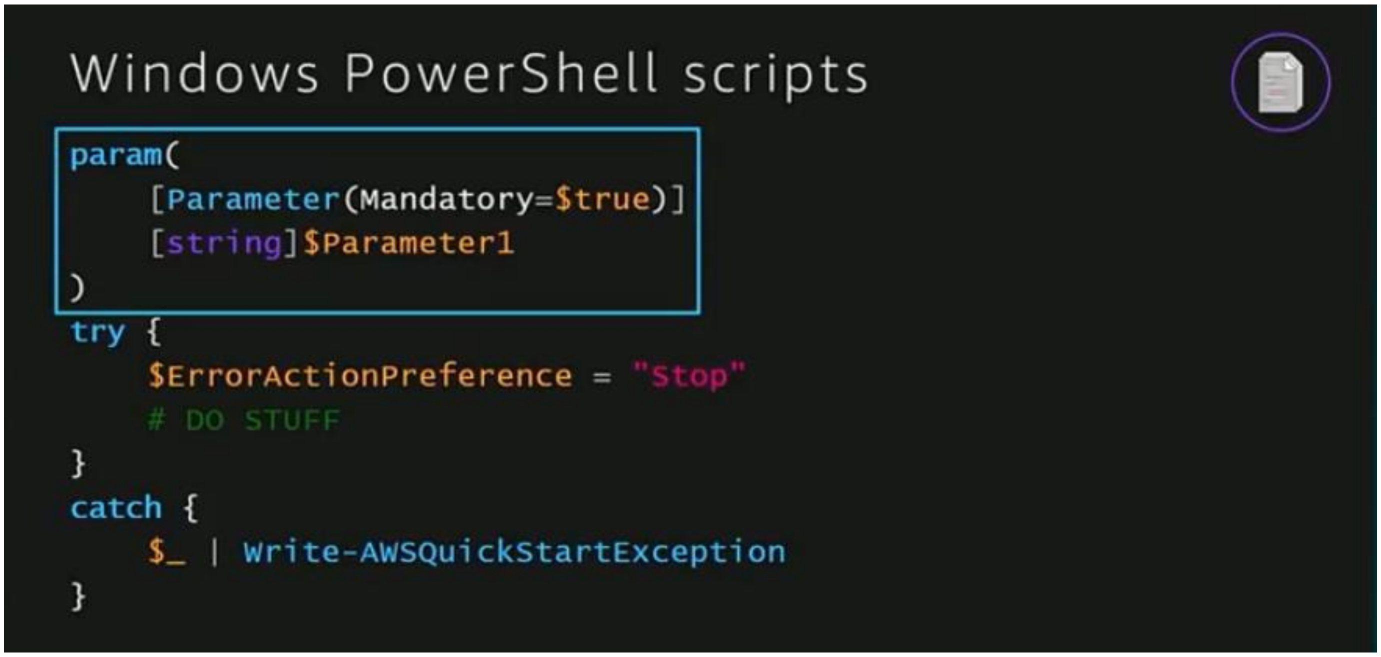
This is how a lot of the signaling within the CloudFormation stack is handled



Windows PowerShell scripts

```
param(
    [Parameter(Mandatory=$true)]
    [string]$Parameter1
)
try {
    $ErrorActionPreference = "Stop"
    # DO STUFF
}
catch {
    $_ | Write-AWSQuickStartException
}
```

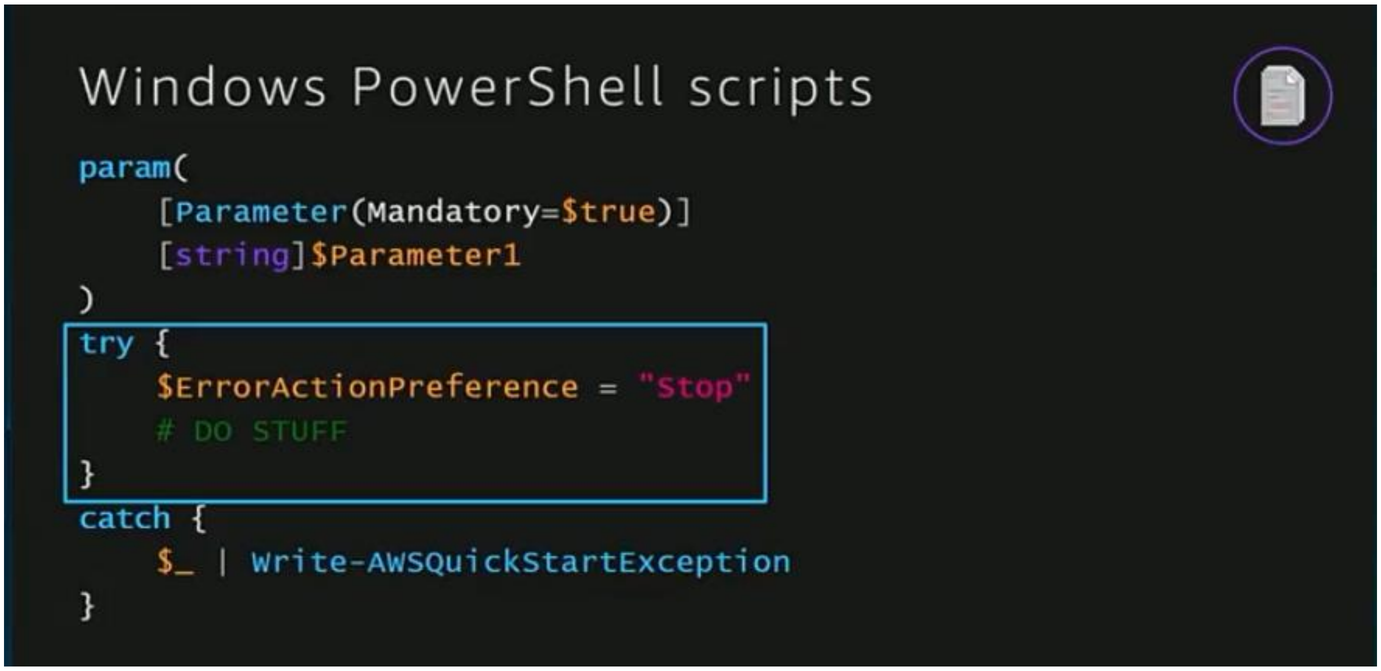
This is a way to build a Powershell script for configuring an instance using CF



Windows PowerShell scripts

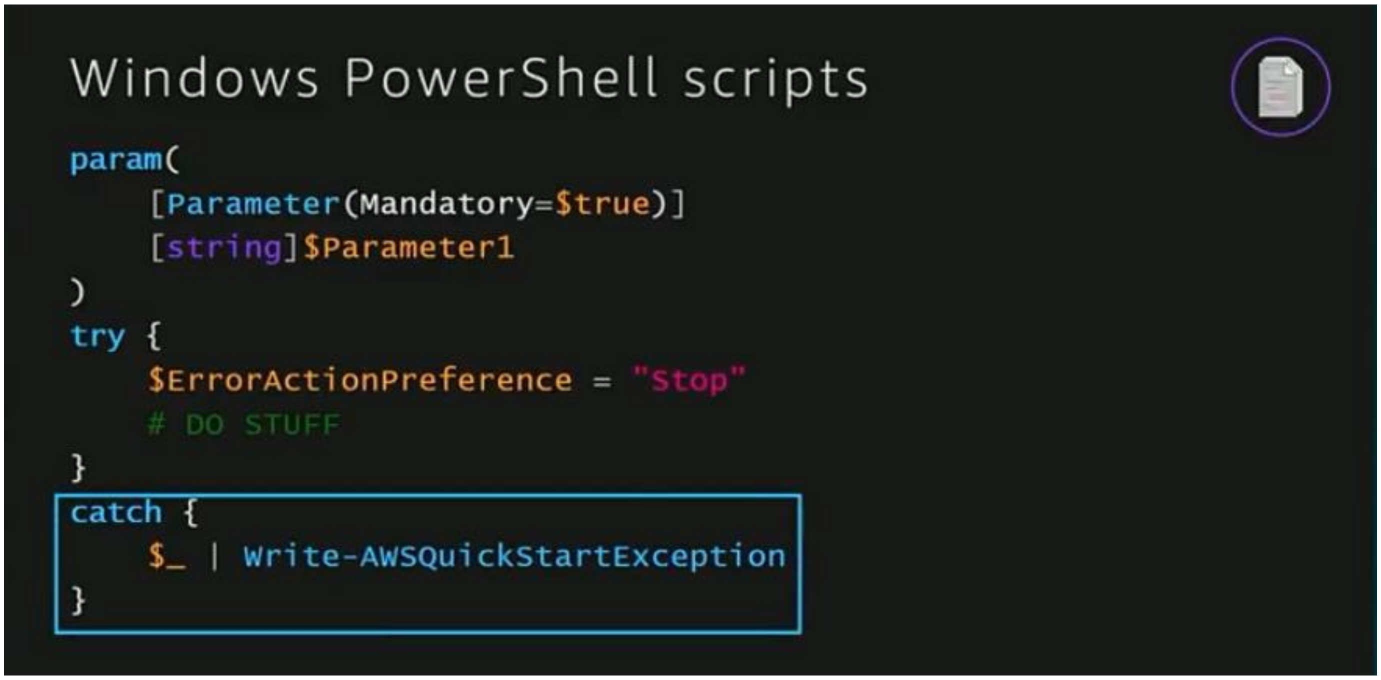
```
param(
    [Parameter(Mandatory=$true)]
    [string]$Parameter1
)
try {
    $ErrorActionPreference = "Stop"
    # DO STUFF
}
catch {
    $_ | Write-AWSQuickStartException
}
```

This starts with the parameter section above; the parameters will be fed in from the CF template



Windows PowerShell scripts

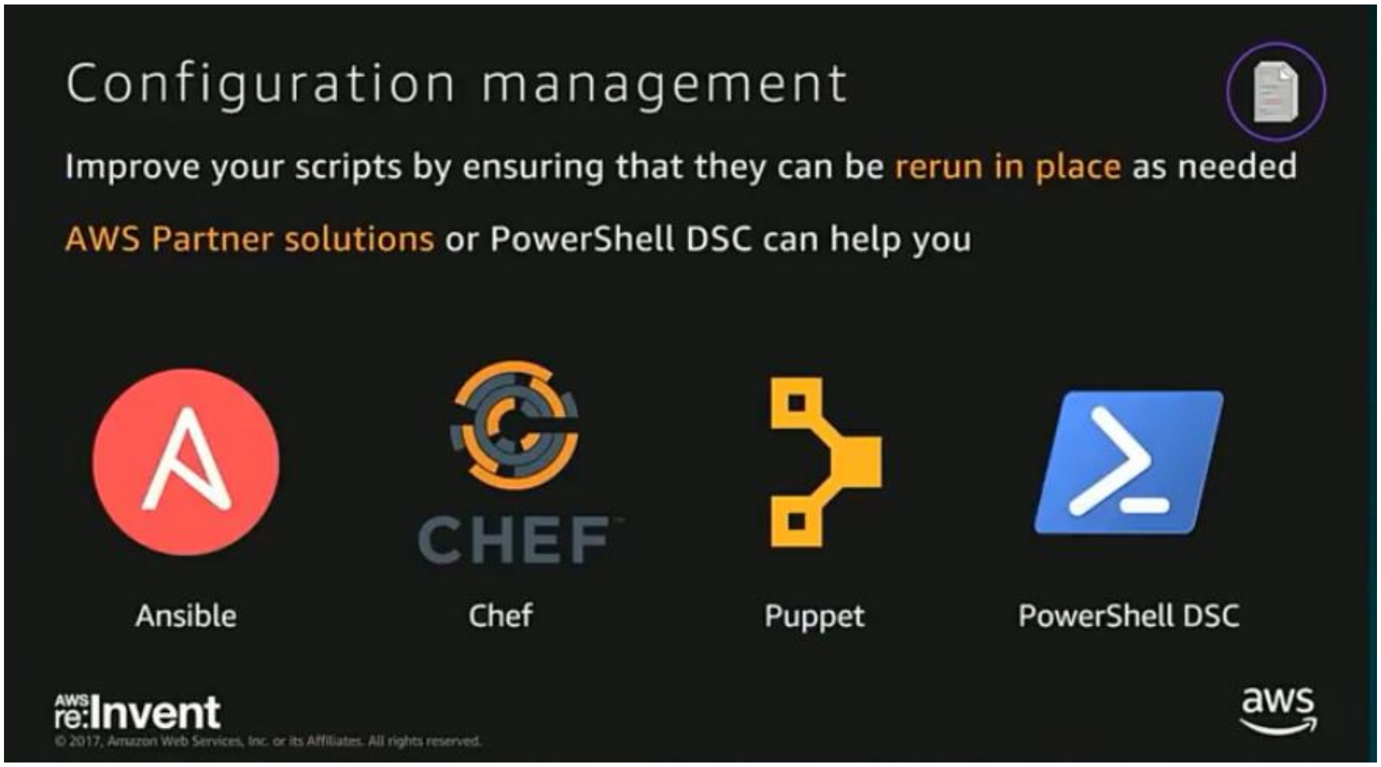
```
param(
    [Parameter(Mandatory=$true)]
    [string]$Parameter1
)
try {
    $ErrorActionPreference = "Stop"
    # DO STUFF
}
catch {
    $_ | Write-AWSQuickStartException
}
```



Windows PowerShell scripts

```
param(
    [Parameter(Mandatory=$true)]
    [string]$Parameter1
)
try {
    $ErrorActionPreference = "Stop"
    # DO STUFF
}
catch {
    $_ | Write-AWSQuickStartException
}
```

The exception is captured and fed to the cmdlet above

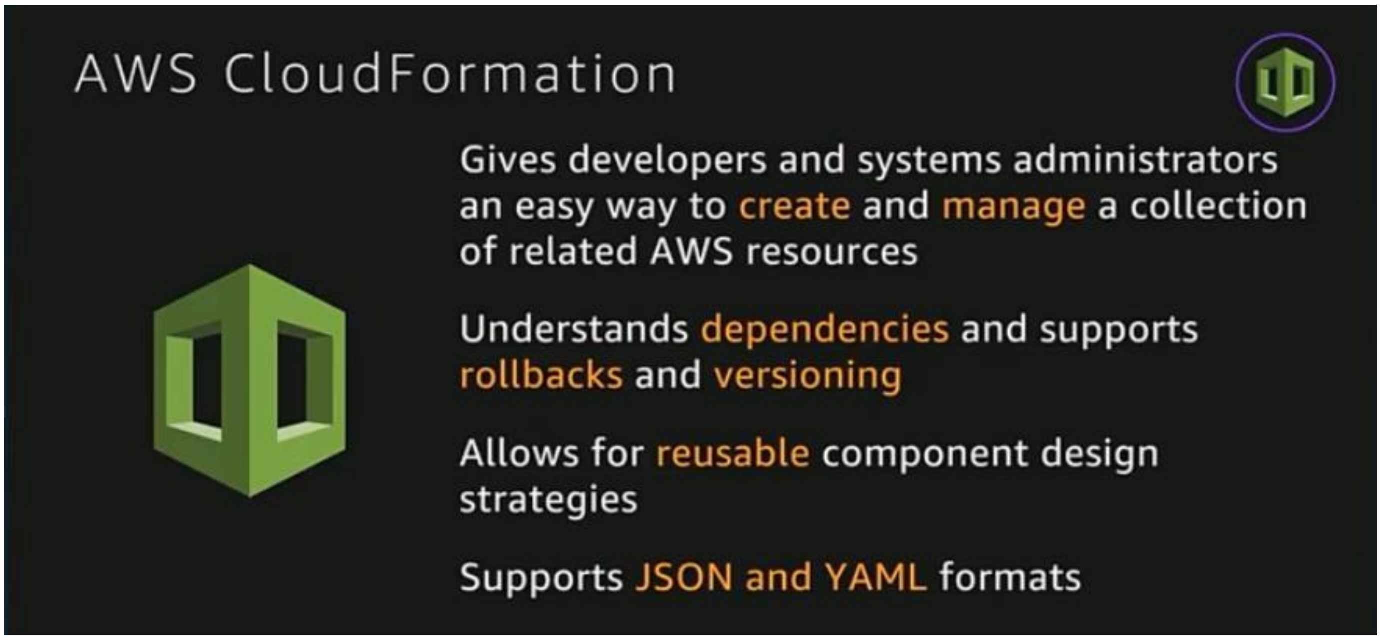


Configuration management

Improve your scripts by ensuring that they can be rerun in place as needed

AWS Partner solutions or PowerShell DSC can help you

Ansible
Chef
Puppet
PowerShell DSC
AWS re:Invent
aws



AWS CloudFormation

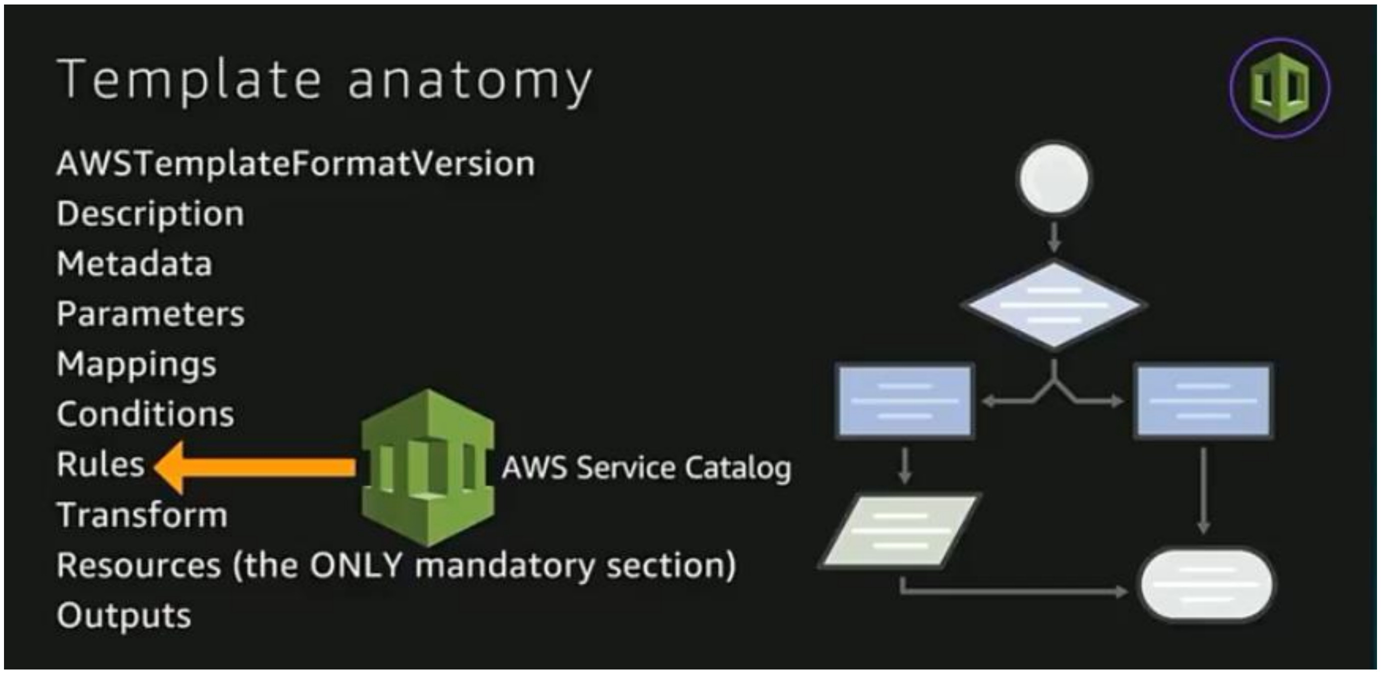
Gives developers and systems administrators an easy way to create and manage a collection of related AWS resources

Understands dependencies and supports rollbacks and versioning

Allows for reusable component design strategies

Supports JSON and YAML formats

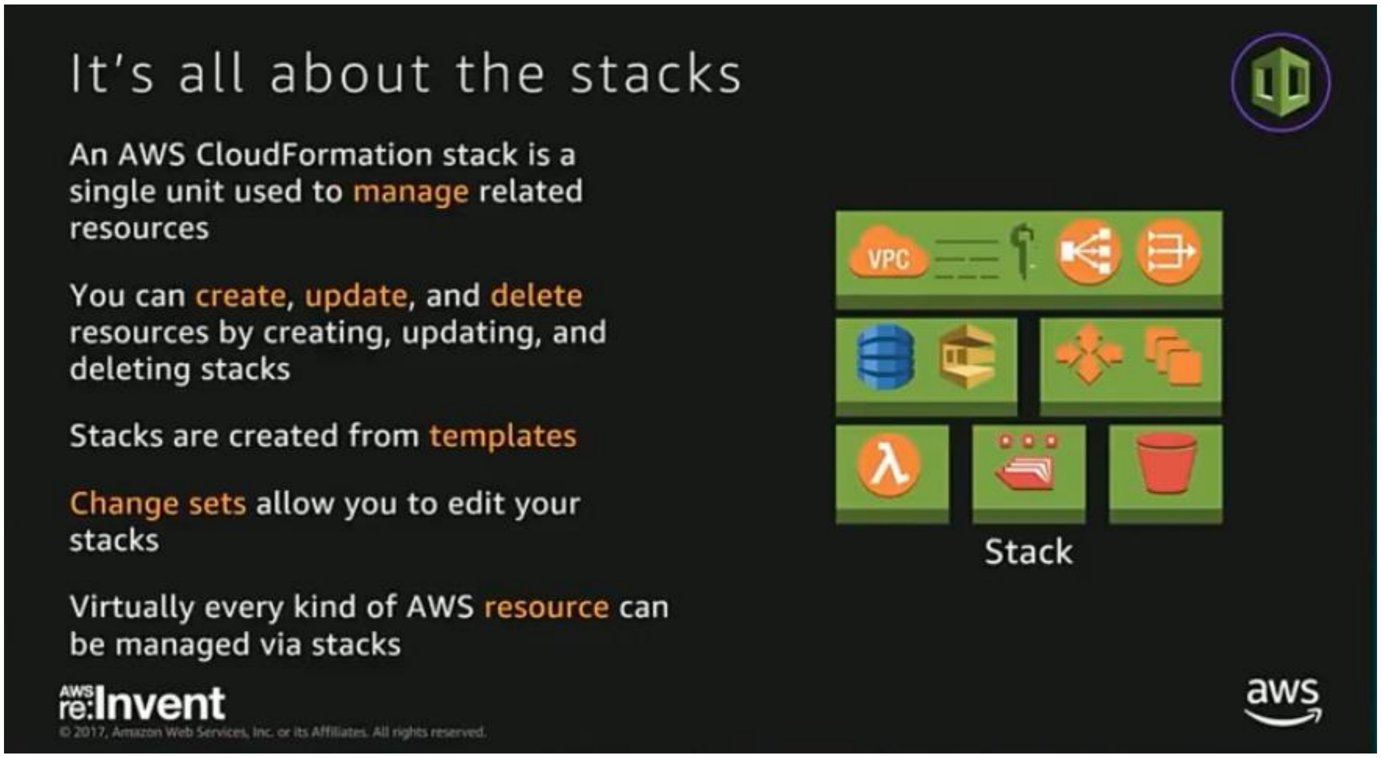
This is the orchestration side, CF gives you a declarative way to create and manage a collection of AWS resources.


Template anatomy

AWSTemplateFormatVersion
Description
Metadata
Parameters
Mappings
Conditions
Rules
Transform
Resources (the ONLY mandatory section)
Outputs

AWS Service Catalog

This is a sample template anatomy for a CF template. Parameters are the inputs, Mappings are the configuration settings that typically are AMI mappings, memory settings, etc. Conditions are conditional statements that get evaluated to determine whether to create some resource, Rules are from the Service Catalog, Rules are template constraints like installing template in a particular region or using a specific instance type only. Transform types include the Serverless Application Model SAM, AWS Include which allows you to pull in another CF snippet into the CF template at execution or deploy time. Outputs are relevant information to be shown to the user.


It's all about the stacks

An AWS CloudFormation stack is a single unit used to manage related resources

You can create, update, and delete resources by creating, updating, and deleting stacks

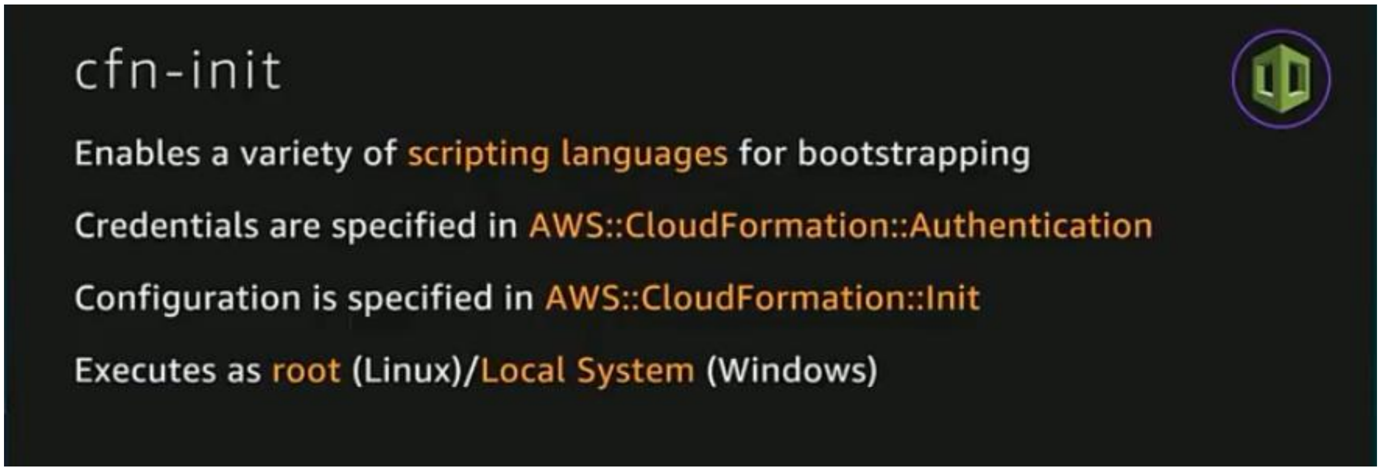
Stacks are created from templates

Change sets allow you to edit your stacks

Virtually every kind of AWS resource can be managed via stacks

VPC
Stack
re:Invent
aws

Change sets create a series of deltas that you can apply to your stack and you can evaluate them before you apply them.



cfn-init

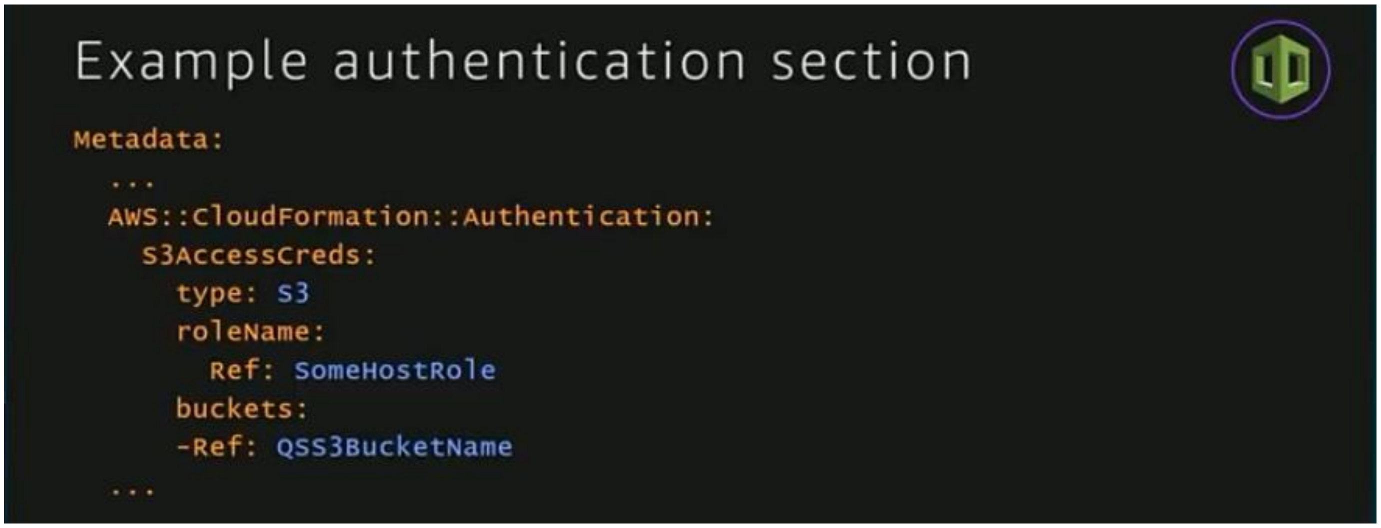
Enables a variety of scripting languages for bootstrapping

Credentials are specified in AWS::CloudFormation::Authentication

Configuration is specified in AWS::CloudFormation::Init

Executes as root (Linux)/Local System (Windows)

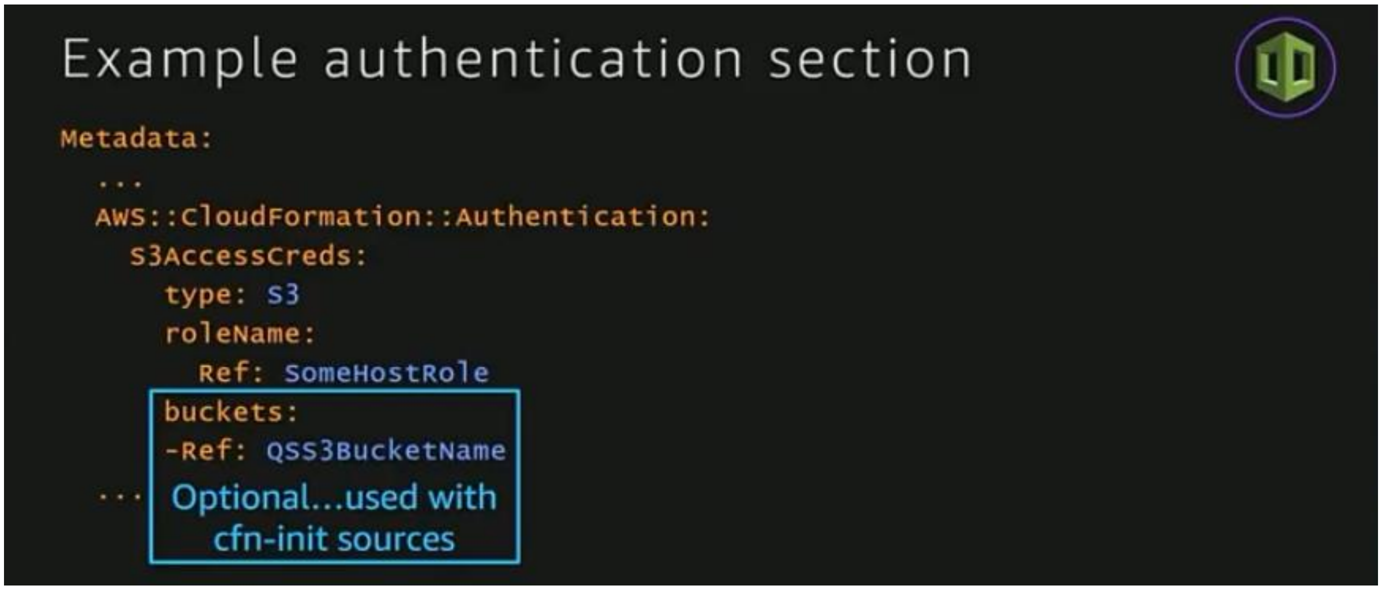
The cfn-init helper script consumes some of the template metadata and enables you to call a variety of scripting languages like shell scripts, PowerShell, Ansible playbooks, Chef recipes, etc.



Example authentication section

```
Metadata:
  ...
  AWS::CloudFormation::Authentication:
    S3AccessCreds:
      type: S3
      roleName:
        Ref: SomeHostRole
      buckets:
      -Ref: QSS3BucketName
  ...
```

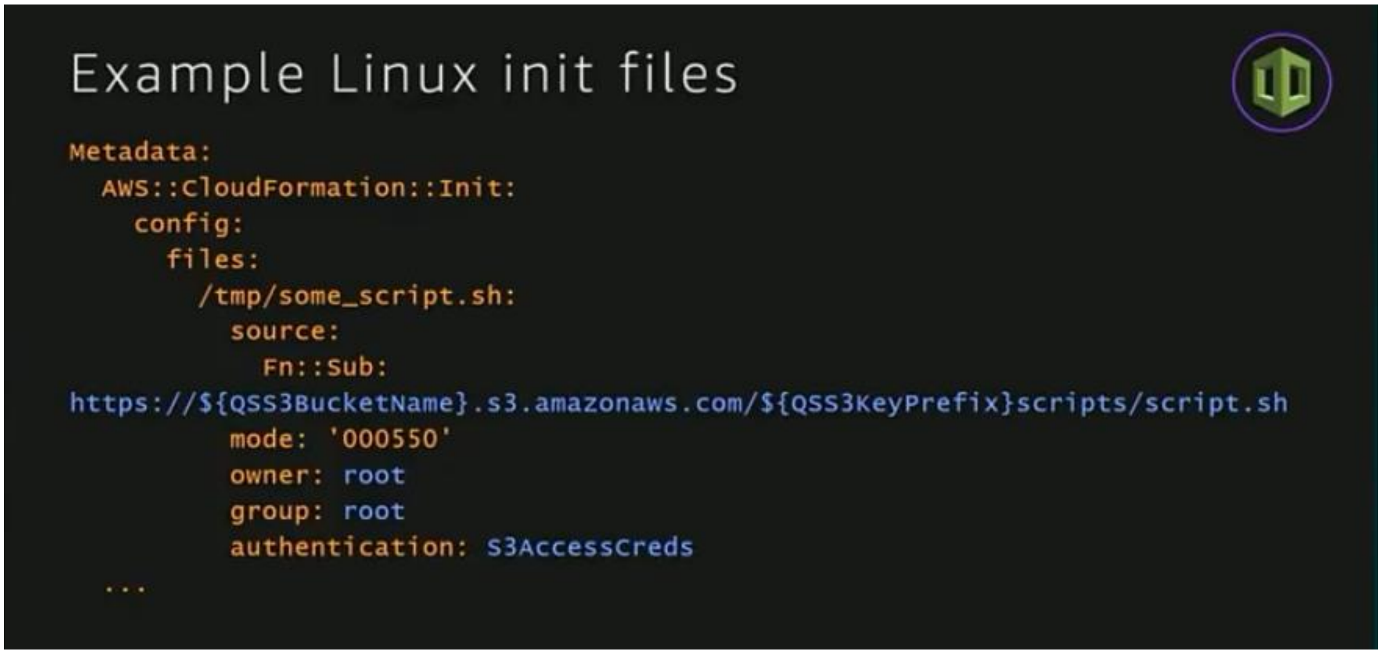
This is an example of an authentication section, these are credentials that we will use to connect to S3, we are also using the roleName of SomeHostRole that is defined in the CF template, we can also specify the buckets that we want to use



Example authentication section

```
Metadata:
  ...
  AWS::CloudFormation::Authentication:
    S3AccessCreds:
      type: S3
      roleName:
        Ref: SomeHostRole
      buckets:
      -Ref: QSS3BucketName
  ...
```

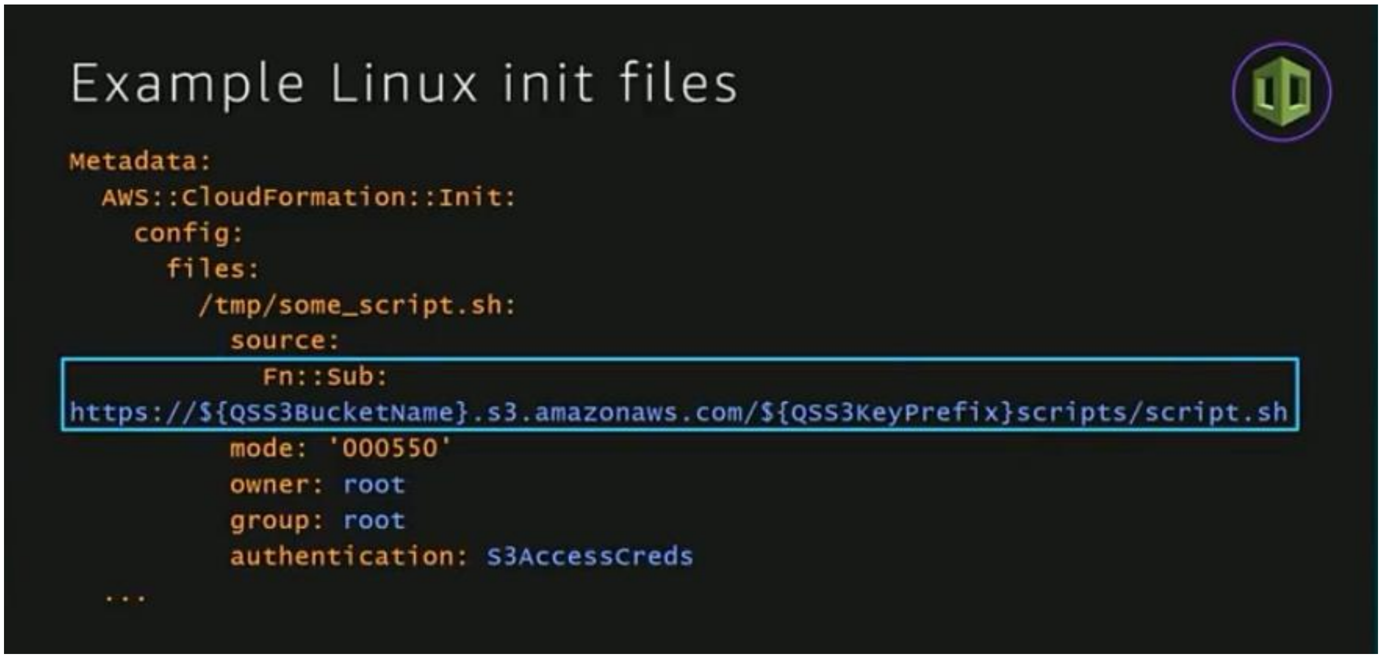
The sources is a section of cfn-init



Example Linux init files

```
Metadata:
  AWS::CloudFormation::Init:
    config:
      files:
        /tmp/some_script.sh:
          source:
            Fn::Sub:
https://${QSS3BucketName}.s3.amazonaws.com/${QSS3KeyPrefix}scripts/script.sh
          mode: '000550'
          owner: root
          group: root
          authentication: S3AccessCreds
  ...
```

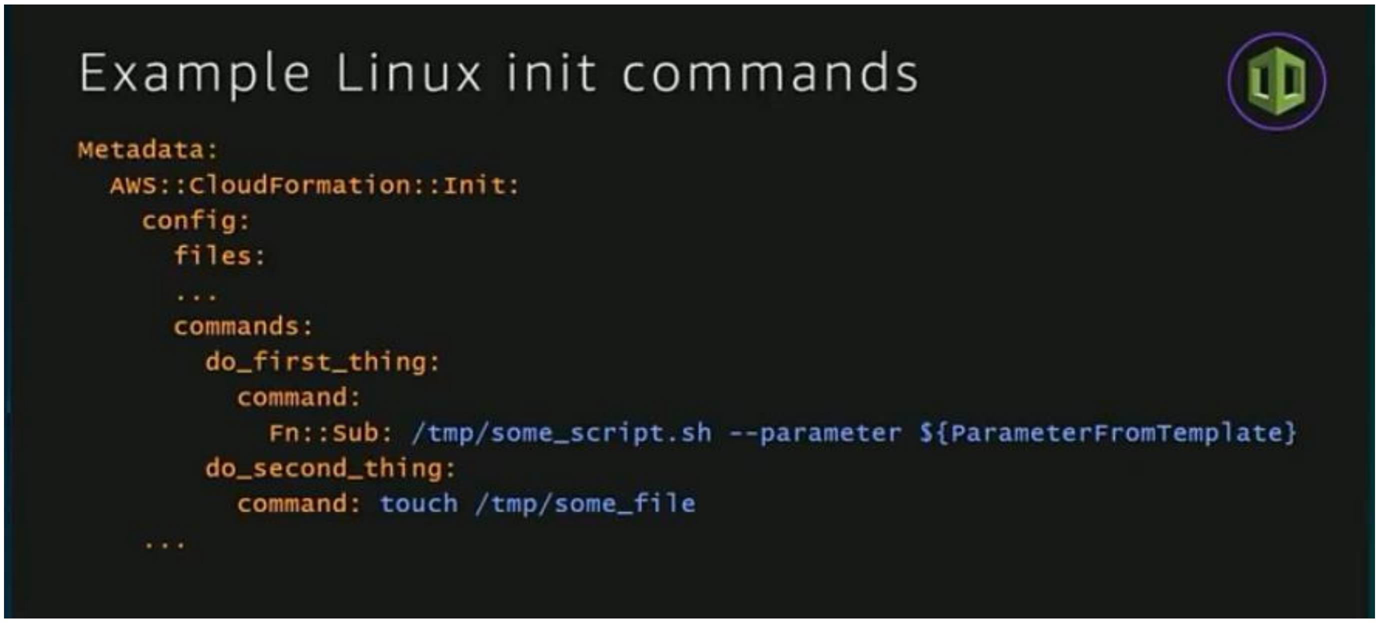
There is a config section, a file section that tells you what the local file name should be.



Example Linux init files

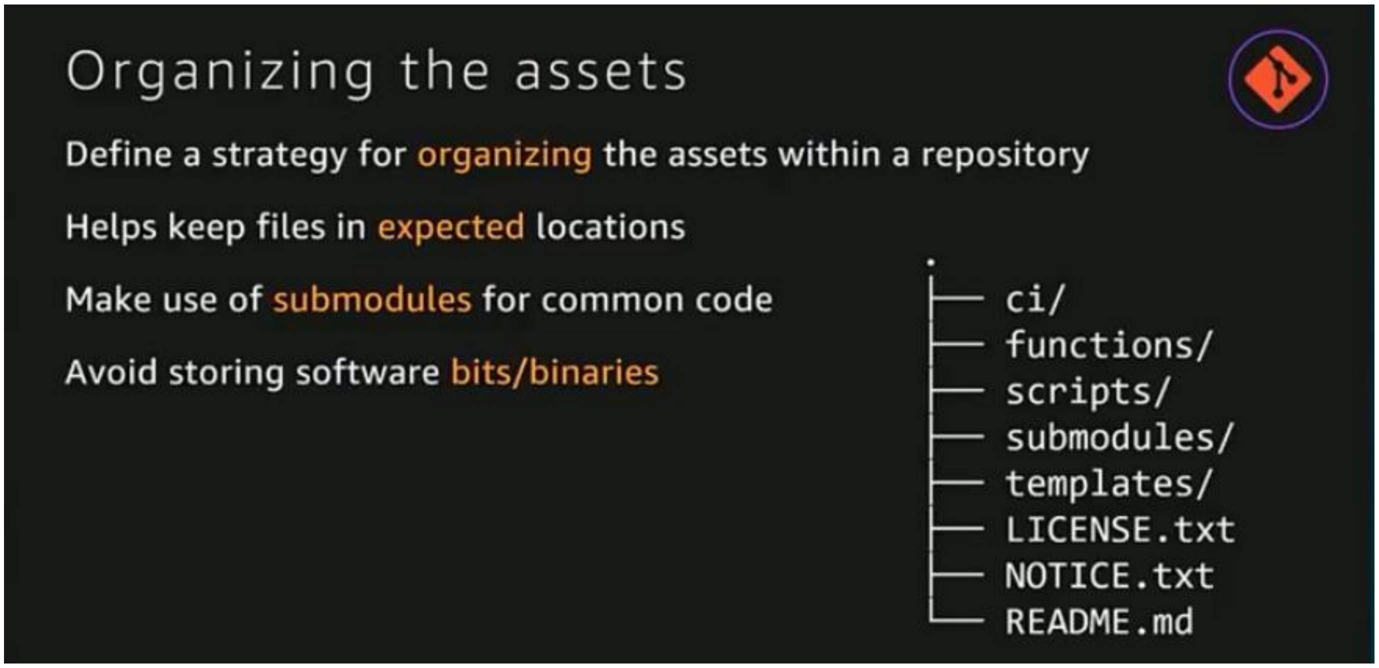
```
Metadata:
  AWS::CloudFormation::Init:
    config:
      files:
        /tmp/some_script.sh:
          source:
            Fn::Sub:
https://${QSS3BucketName}.s3.amazonaws.com/${QSS3KeyPrefix}scripts/script.sh
          mode: '000550'
          owner: root
          group: root
          authentication: S3AccessCreds
  ...
```

We then use the substitution function ***Fn::Sub*** to dynamically build a URL using some parameters from the template and getting the script.sh from that URL. Then we dynamically set the mode, owner, and group permission on the file and use the authentication credentials from some other part of the template



Example Linux init commands

```yaml
Metadata:
  AWS::CloudFormation::Init:
    config:
      files:
      ...
      commands:
        do_first_thing:
          command:
            Fn::Sub: /tmp/some_script.sh --parameter ${ParameterFromTemplate}
        do_second_thing:
          command: touch /tmp/some_file
    ...
```



Organizing the assets

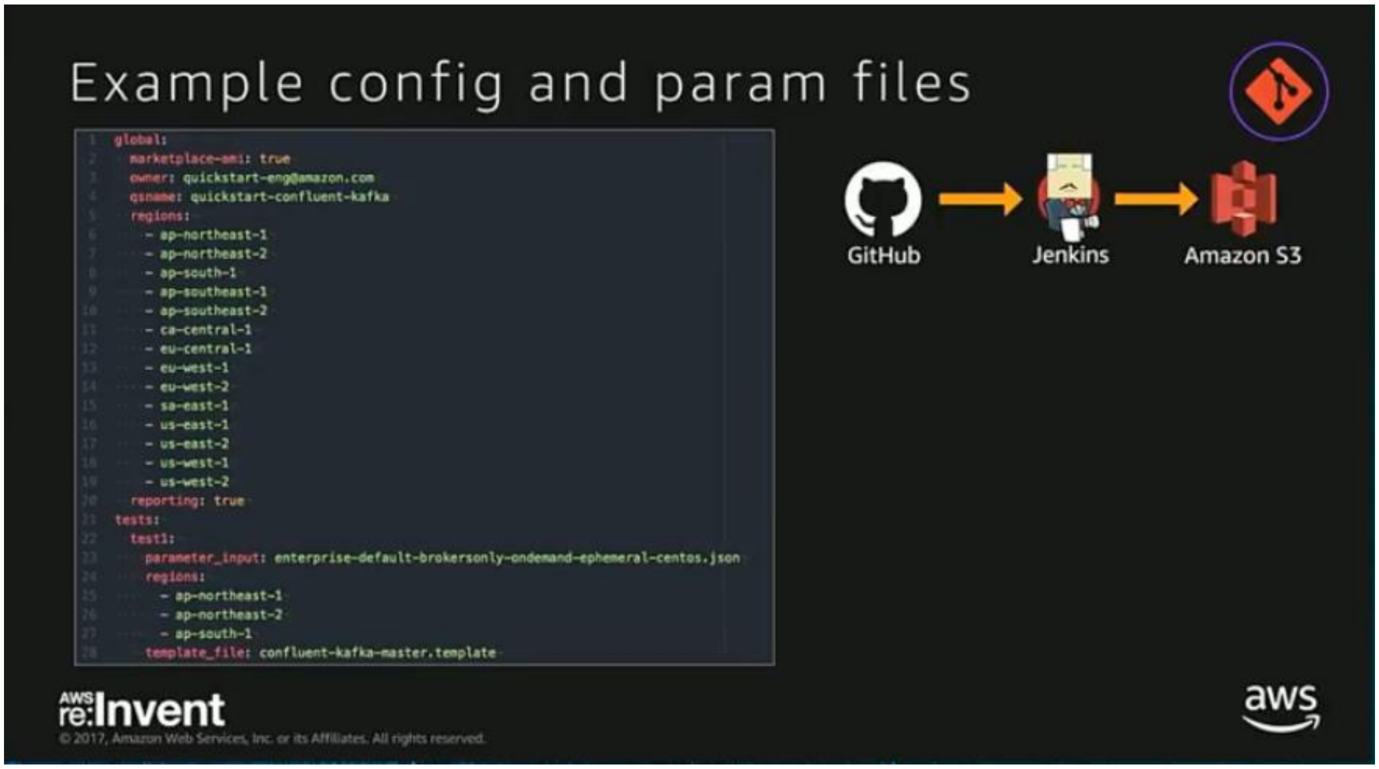
Define a strategy for organizing the assets within a repository

Helps keep files in expected locations

Make use of submodules for common code

Avoid storing software bits/binaries

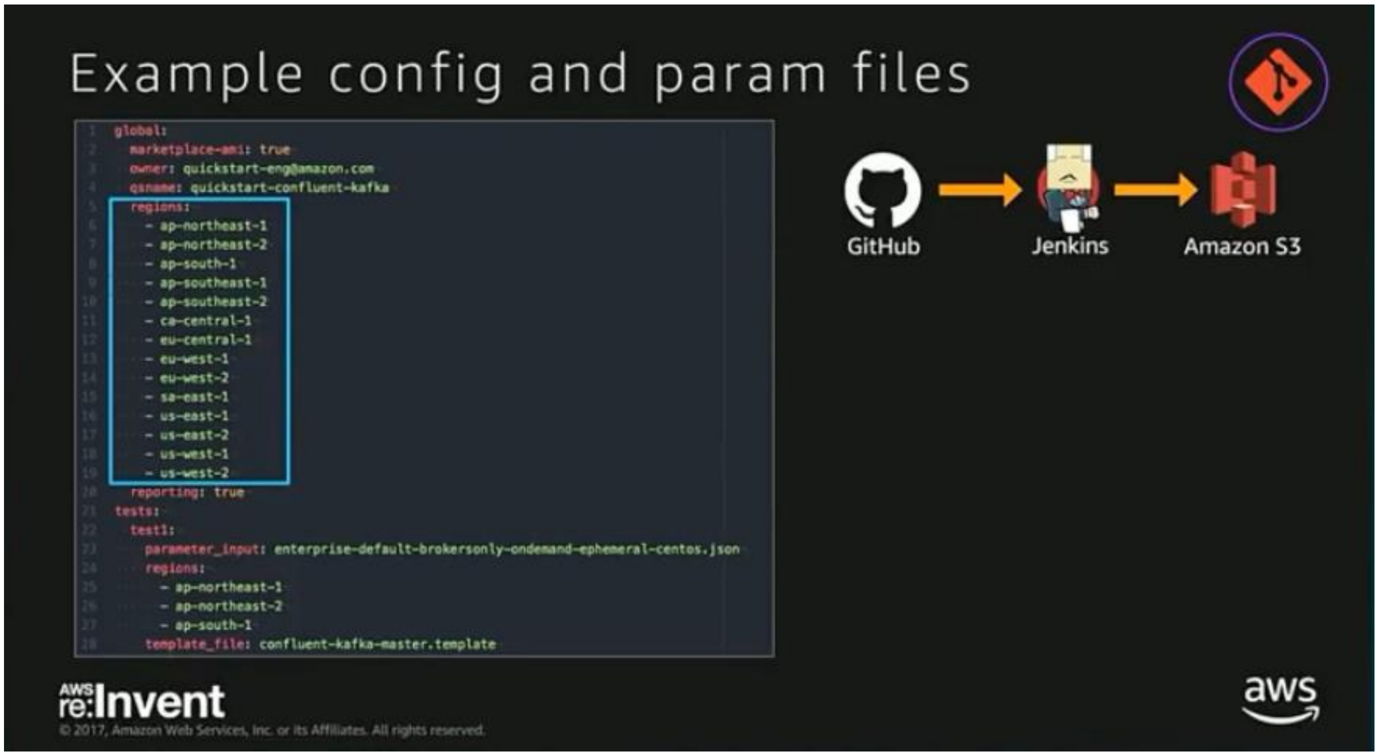
```
.
├── ci/
├── functions/
├── scripts/
├── submodules/
├── templates/
├── LICENSE.txt
├── NOTICE.txt
└── README.md
```



Example config and param files

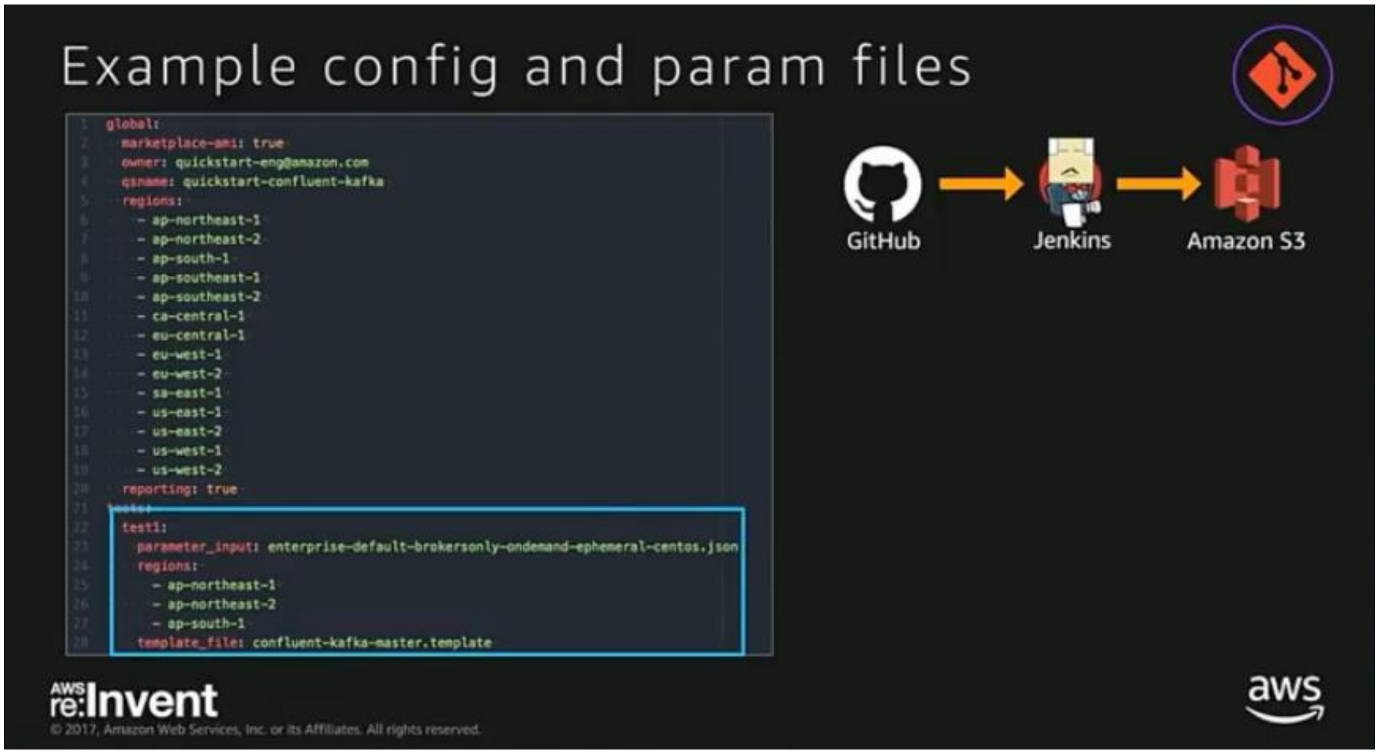
```yaml
global:
  marketplace-ami: true
  owner: quickstart-eng@amazon.com
  qsname: quickstart-confluent-kafka
  regions:
    - ap-northeast-1
    - ap-northeast-2
    - ap-south-1
    - ap-southeast-1
    - ap-southeast-2
    - ca-central-1
    - eu-central-1
    - eu-west-1
    - eu-west-2
    - sa-east-1
    - us-east-1
    - us-east-2
    - us-west-1
    - us-west-2
  reporting: true
tests:
  test1:
    parameter_input: enterprise-default-brokersonly-ondemand-ephemeral-centos.json
    regions:
      - ap-northeast-1
      - ap-northeast-2
      - ap-south-1
    template_file: confluent-kafka-master.template
```

We use GitHub and have Jenkins servers that are watching the GitHub branches for changes to run some tests and then publish artifacts into S3.



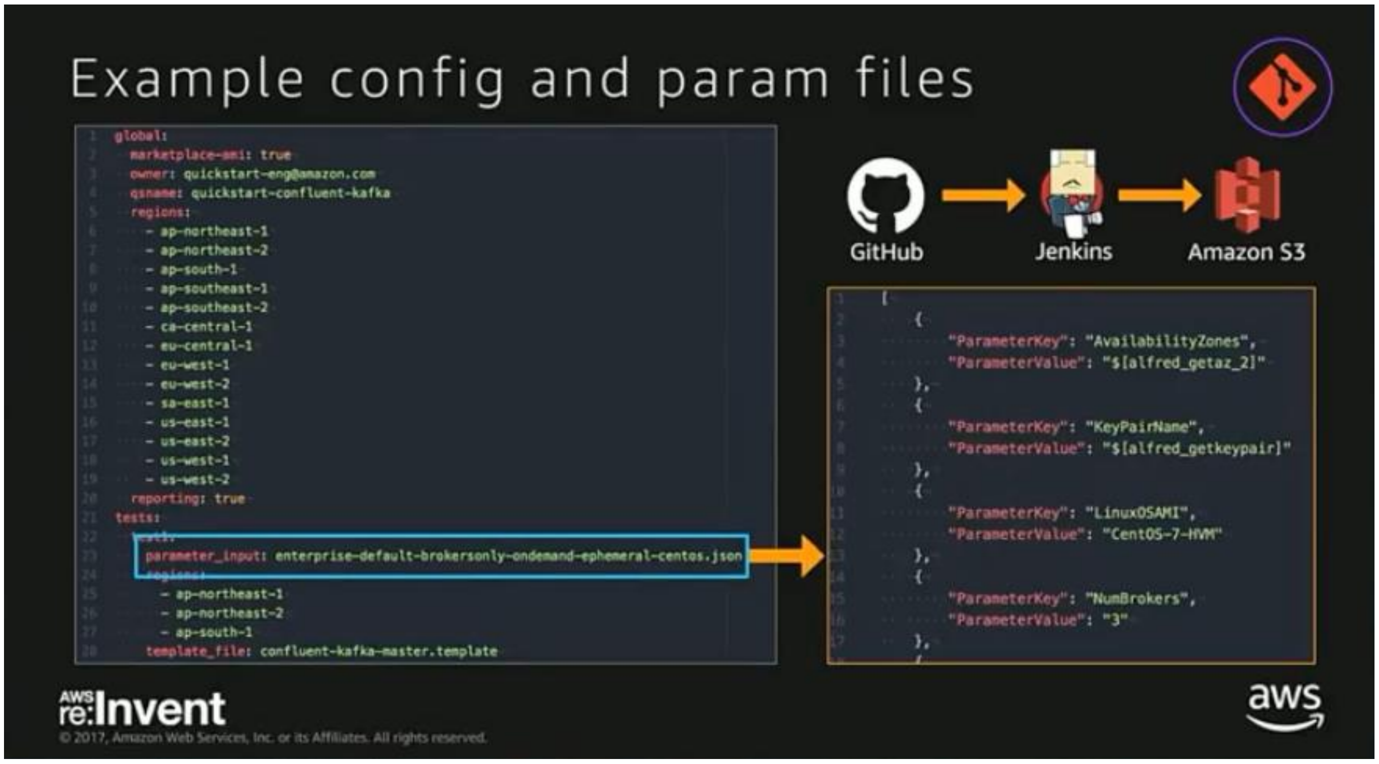
Example config and param files

```yaml
global:
  marketplace-ami: true
  owner: quickstart-eng@amazon.com
  qsname: quickstart-confluent-kafka
  regions:
    - ap-northeast-1
    - ap-northeast-2
    - ap-south-1
    - ap-southeast-1
    - ap-southeast-2
    - ca-central-1
    - eu-central-1
    - eu-west-1
    - eu-west-2
    - sa-east-1
    - us-east-1
    - us-east-2
    - us-west-1
    - us-west-2
  reporting: true
tests:
  test1:
    parameter_input: enterprise-default-brokersonly-ondemand-ephemeral-centos.json
    regions:
      - ap-northeast-1
      - ap-northeast-2
      - ap-south-1
    template_file: confluent-kafka-master.template
```



Example config and param files

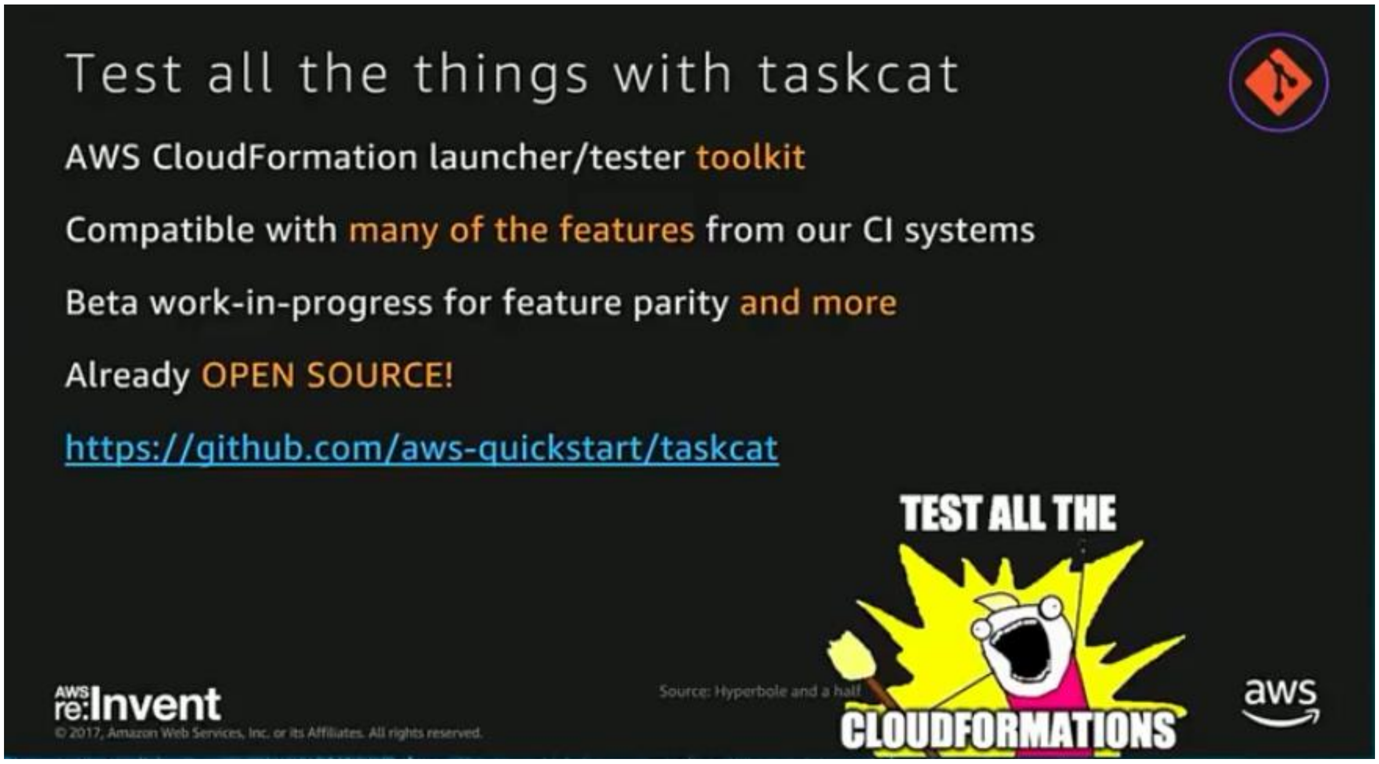
```yaml
global:
  marketplace-ami: true
  owner: quickstart-eng@amazon.com
  qsname: quickstart-confluent-kafka
  regions:
    - ap-northeast-1
    - ap-northeast-2
    - ap-south-1
    - ap-southeast-1
    - ap-southeast-2
    - ca-central-1
    - eu-central-1
    - eu-west-1
    - eu-west-2
    - sa-east-1
    - us-east-1
    - us-east-2
    - us-west-1
    - us-west-2
  reporting: true
tests:
  test1:
    parameter_input: enterprise-default-brokersonly-ondemand-ephemeral-centos.json
    regions:
      - ap-northeast-1
      - ap-northeast-2
      - ap-south-1
    template_file: confluent-kafka-master.template
```



Example config and param files

```yaml
global:
  marketplace-ami: true
  owner: quickstart-eng@amazon.com
  qsname: quickstart-confluent-kafka
  regions:
    - ap-northeast-1
    - ap-northeast-2
    - ap-south-1
    - ap-southeast-1
    - ap-southeast-2
    - ca-central-1
    - eu-central-1
    - eu-west-1
    - eu-west-2
    - sa-east-1
    - us-east-1
    - us-east-2
    - us-west-1
    - us-west-2
  reporting: true
tests:
  test1:
    parameter_input: enterprise-default-brokersonly-ondemand-ephemeral-centos.json
    regions:
      - ap-northeast-1
      - ap-northeast-2
      - ap-south-1
    template_file: confluent-kafka-master.template
```

```json
[
    {
        "ParameterKey": "AvailabilityZones",
        "ParameterValue": "$[alfred_getaz_2]"
    },
    {
        "ParameterKey": "KeyPairName",
        "ParameterValue": "$[alfred_getkeypair]"
    },
    {
        "ParameterKey": "LinuxOSAMI",
        "ParameterValue": "CentOS-7-HVM"
    },
    {
        "ParameterKey": "NumBrokers",
        "ParameterValue": "3"
    },
```



Test all the things with taskcat

AWS CloudFormation launcher/tester toolkit

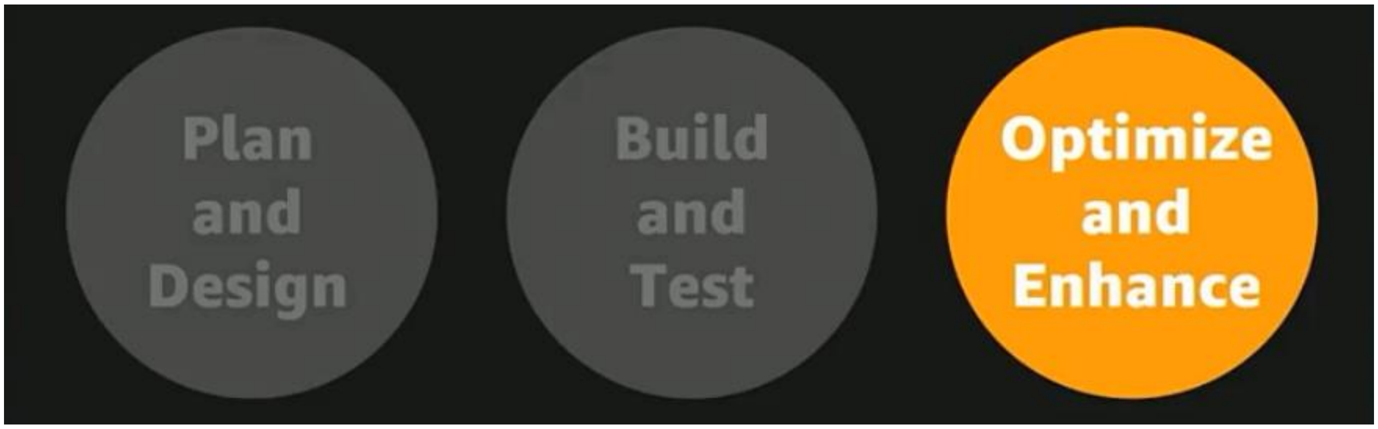
Compatible with many of the features from our CI systems

Beta work-in-progress for feature parity and more

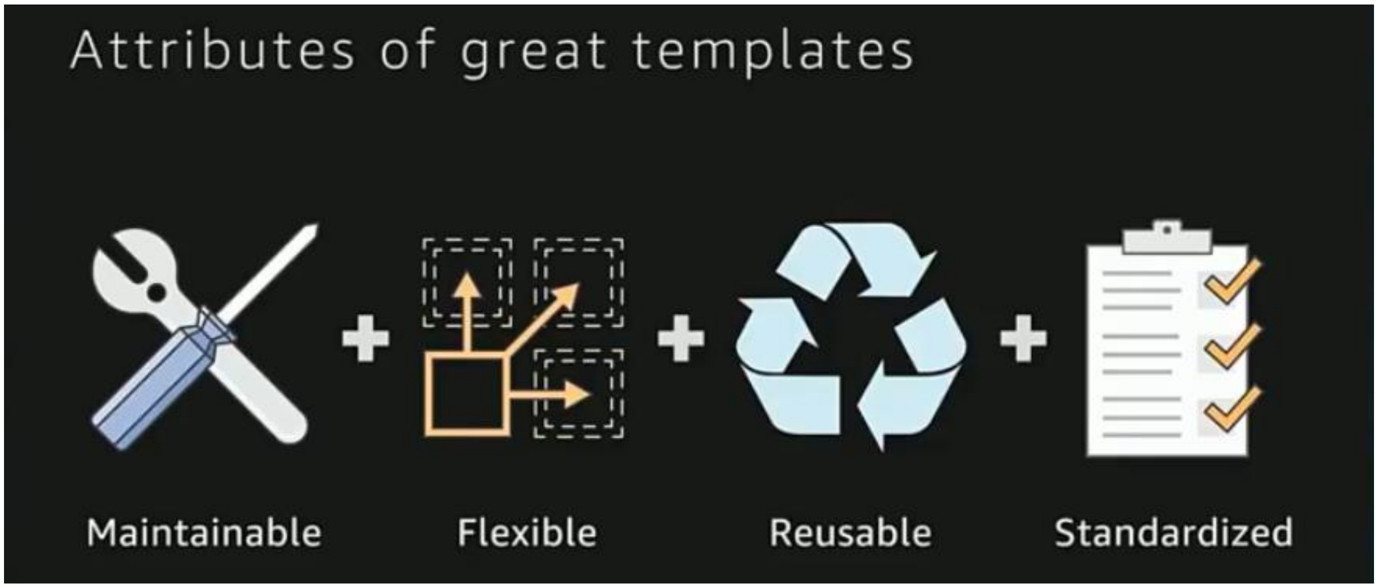
Already OPEN SOURCE!

https://github.com/aws-quickstart/taskcat

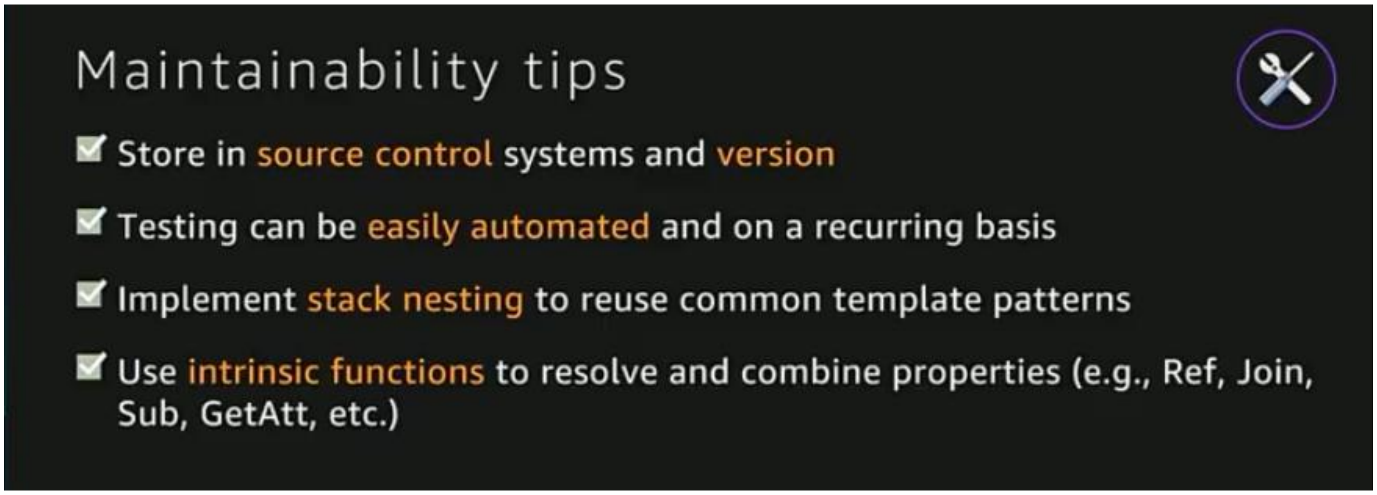
TEST ALL THE
CLOUDFORMATIONS
Source: Hyperbole and a half
AWS re:Invent
© 2017, Amazon Web Services, Inc. or its Affiliates. All rights reserved.
aws



This is the last phase of development

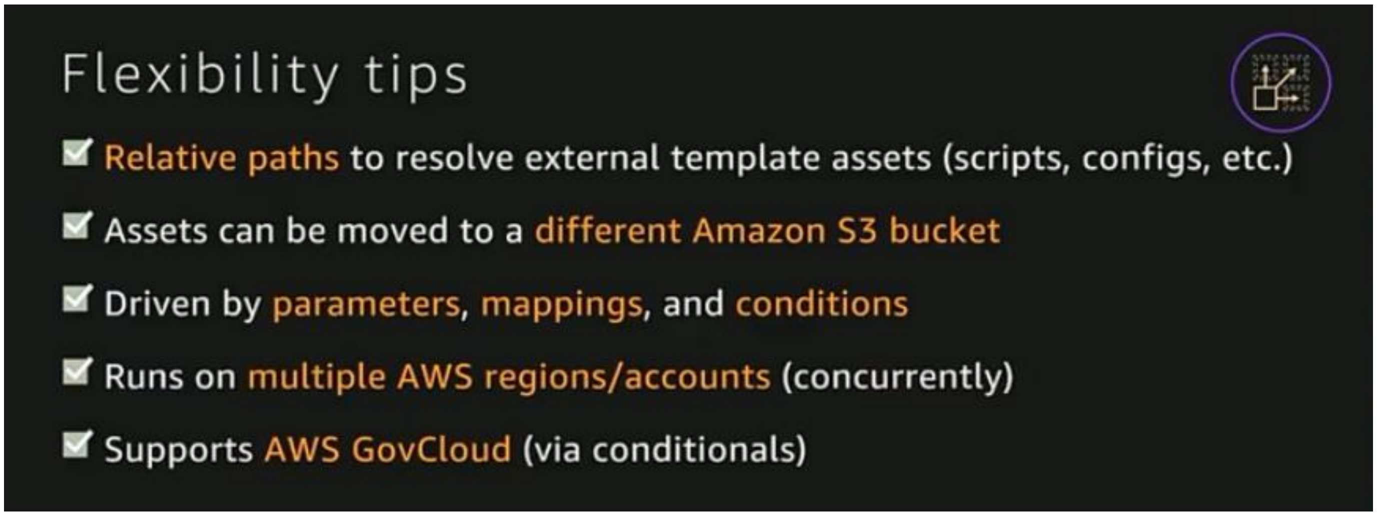


Attributes of great templates



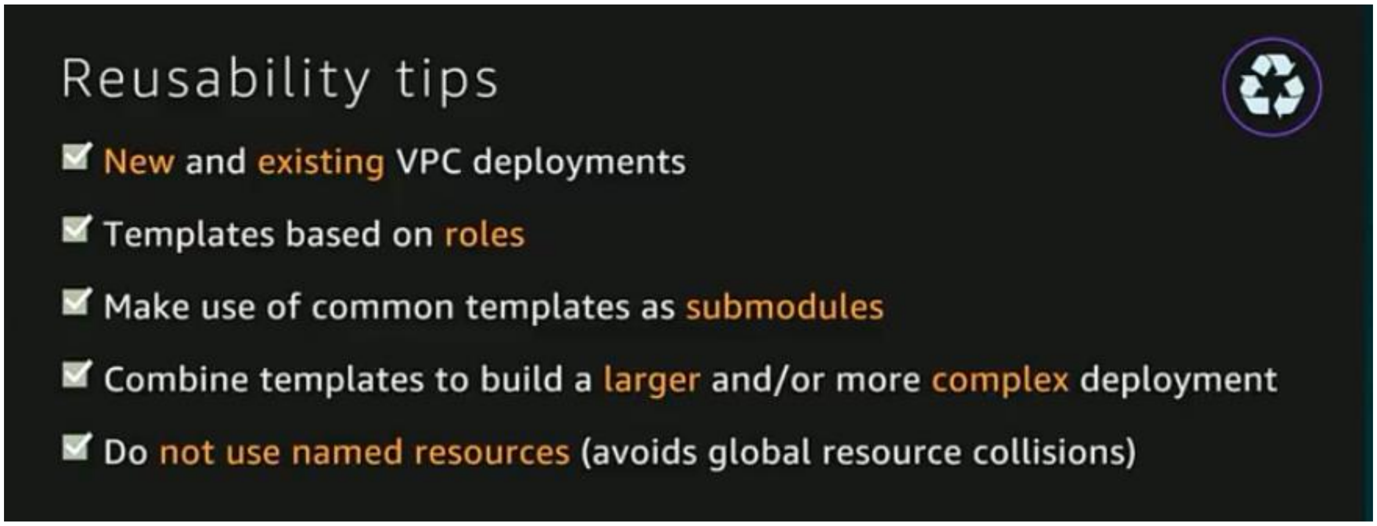
Maintainability tips

- Store in source control systems and version
- Testing can be easily automated and on a recurring basis
- Implement stack nesting to reuse common template patterns
- Use intrinsic functions to resolve and combine properties (e.g., Ref, Join, Sub, GetAtt, etc.)



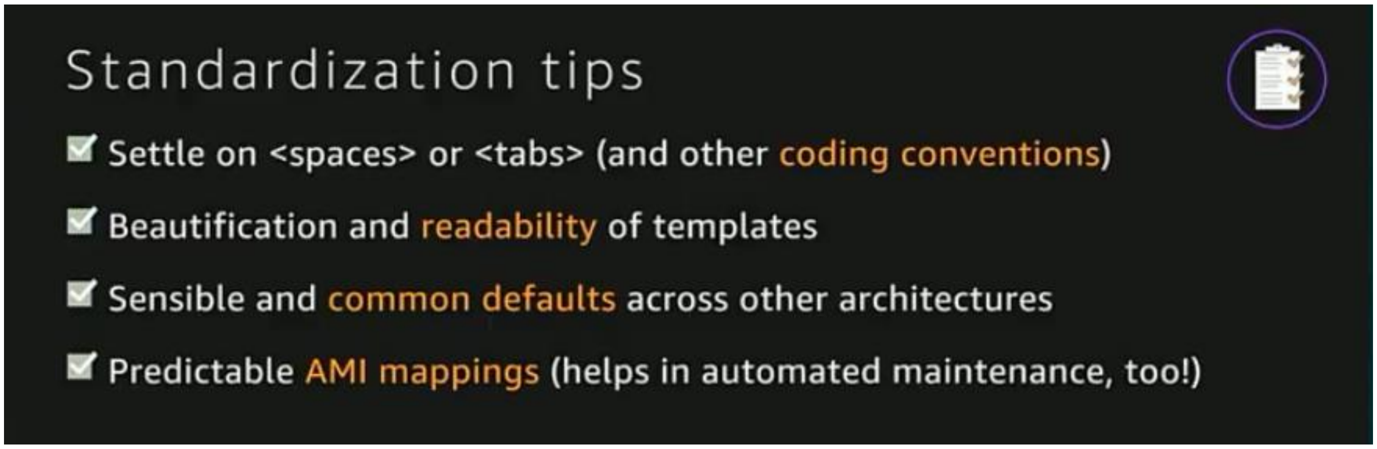
Flexibility tips

- Relative paths to resolve external template assets (scripts, configs, etc.)
- Assets can be moved to a different Amazon S3 bucket
- Driven by parameters, mappings, and conditions
- Runs on multiple AWS regions/accounts (concurrently)
- Supports AWS GovCloud (via conditionals)



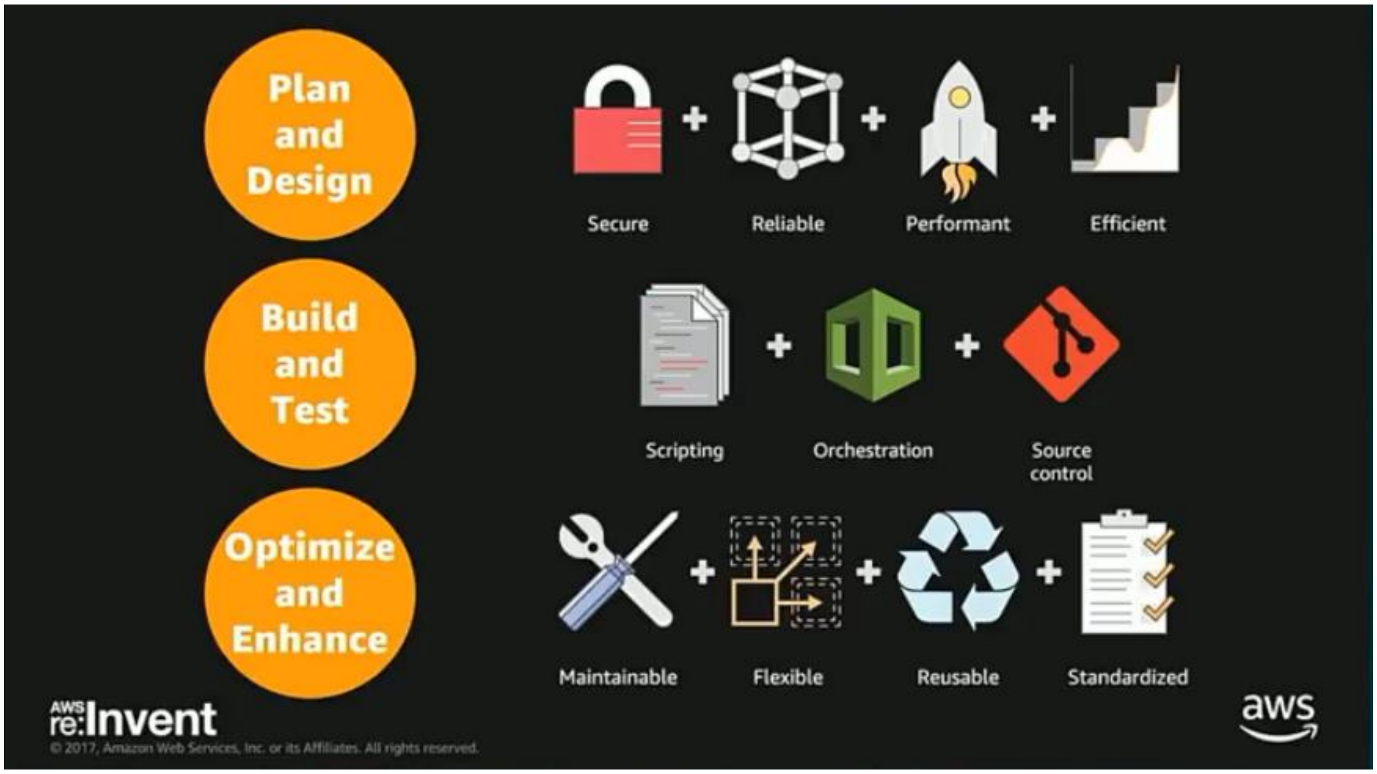
Reusability tips

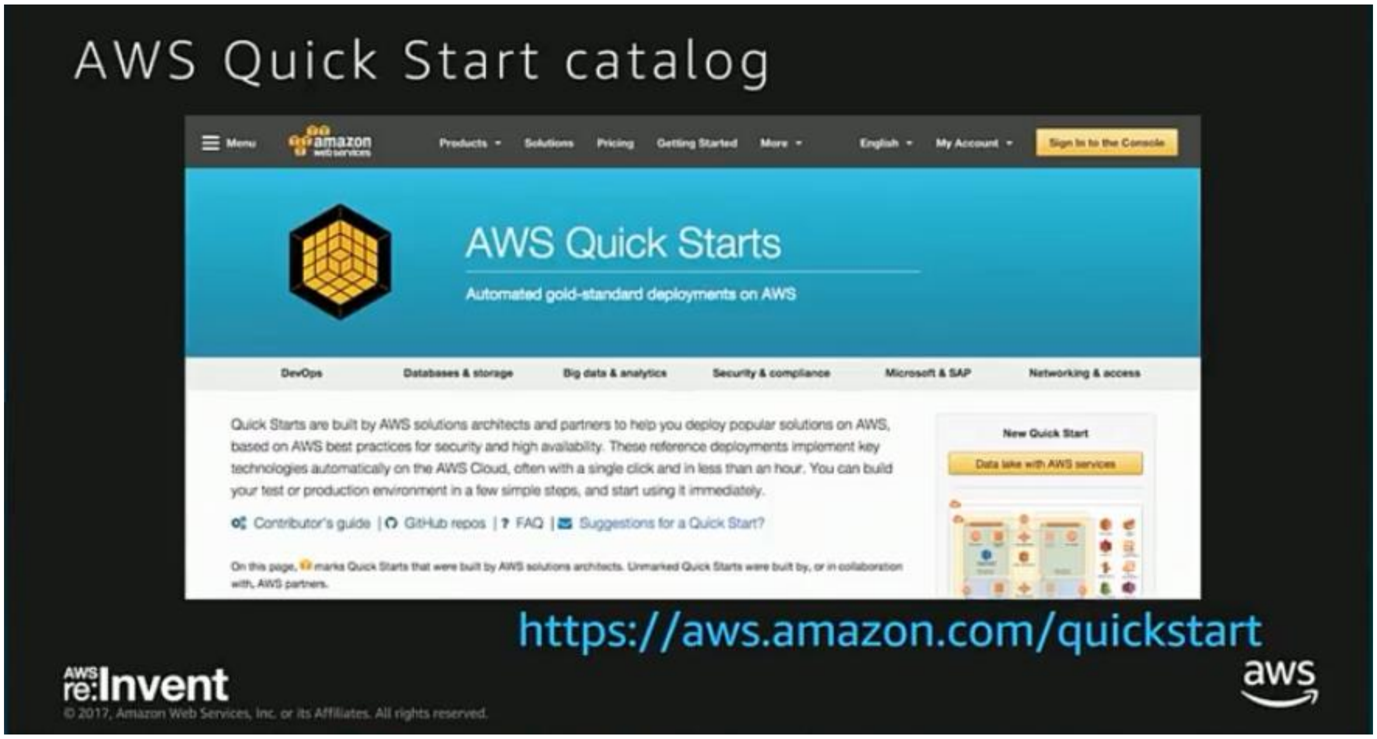
- New and existing VPC deployments
- Templates based on roles
- Make use of common templates as submodules
- Combine templates to build a larger and/or more complex deployment
- Do not use named resources (avoids global resource collisions)



Standardization tips

- ✓ Settle on <spaces> or <tabs> (and other coding conventions)
- ✓ Beautification and readability of templates
- ✓ Sensible and common defaults across other architectures
- ✓ Predictable AMI mappings (helps in automated maintenance, too!)





AWS Quick Start catalog

AWS Quick Starts

Automated gold-standard deployments on AWS

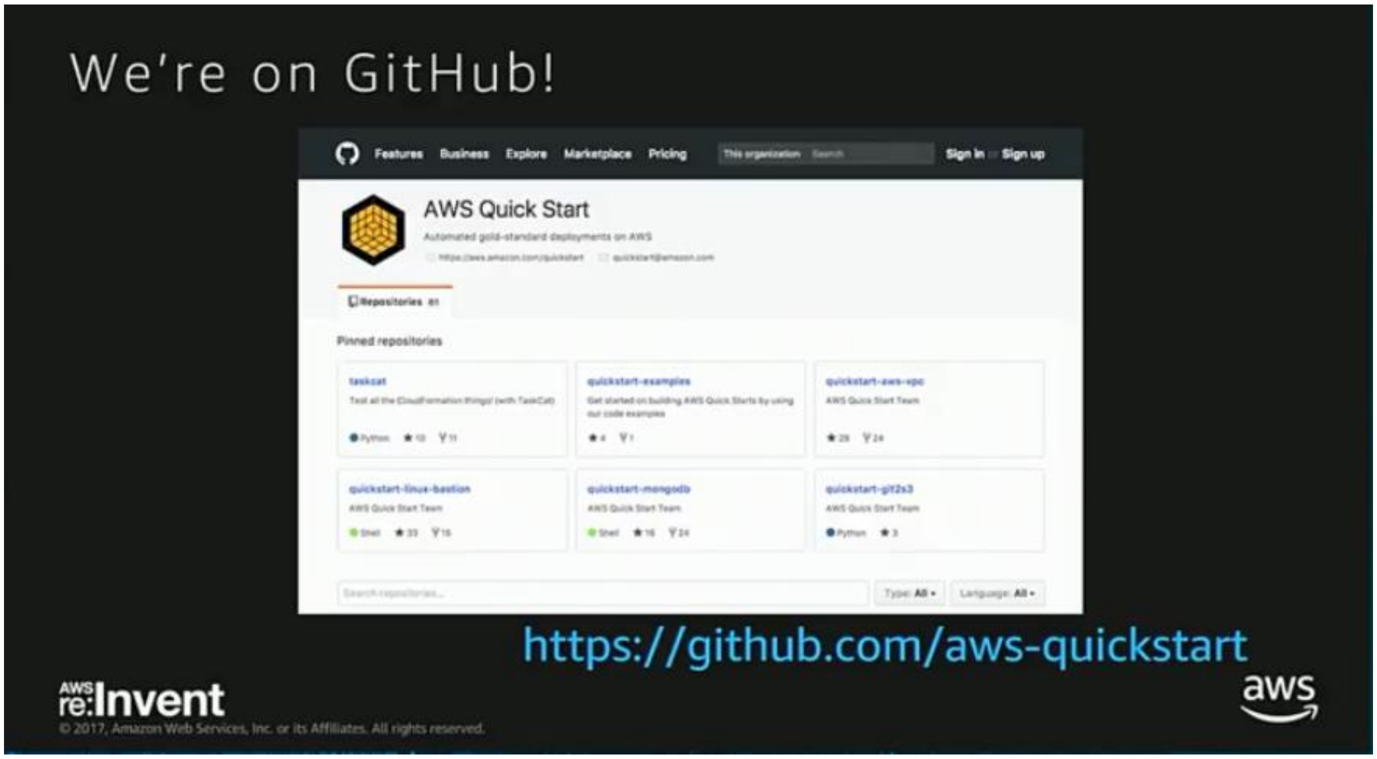
Quick Starts are built by AWS solutions architects and partners to help you deploy popular solutions on AWS, based on AWS best practices for security and high availability. These reference deployments implement key technologies automatically on the AWS Cloud, often with a single click and in less than an hour. You can build your test or production environment in a few simple steps, and start using it immediately.

Contributor's guide | GitHub repos | ? FAQ | Suggestions for a Quick Start?
New Quick Start
Data lake with AWS services

On this page, marks Quick Starts that were built by AWS solutions architects. Unmarked Quick Starts were built by, or in collaboration with, AWS partners.

https://aws.amazon.com/quickstart

There are available workloads that you can leverage right away



We're on GitHub!

AWS Quick Start

Automated gold-standard deployments on AWS

https://aws.amazon.com/quickstart quickstart@amazon.com

taskcat
Test all the CloudFormation things! (with TaskCat)
Python ★ 10 11

quickstart-examples
Get started on building AWS Quick Starts by using our code examples
★ 4 1

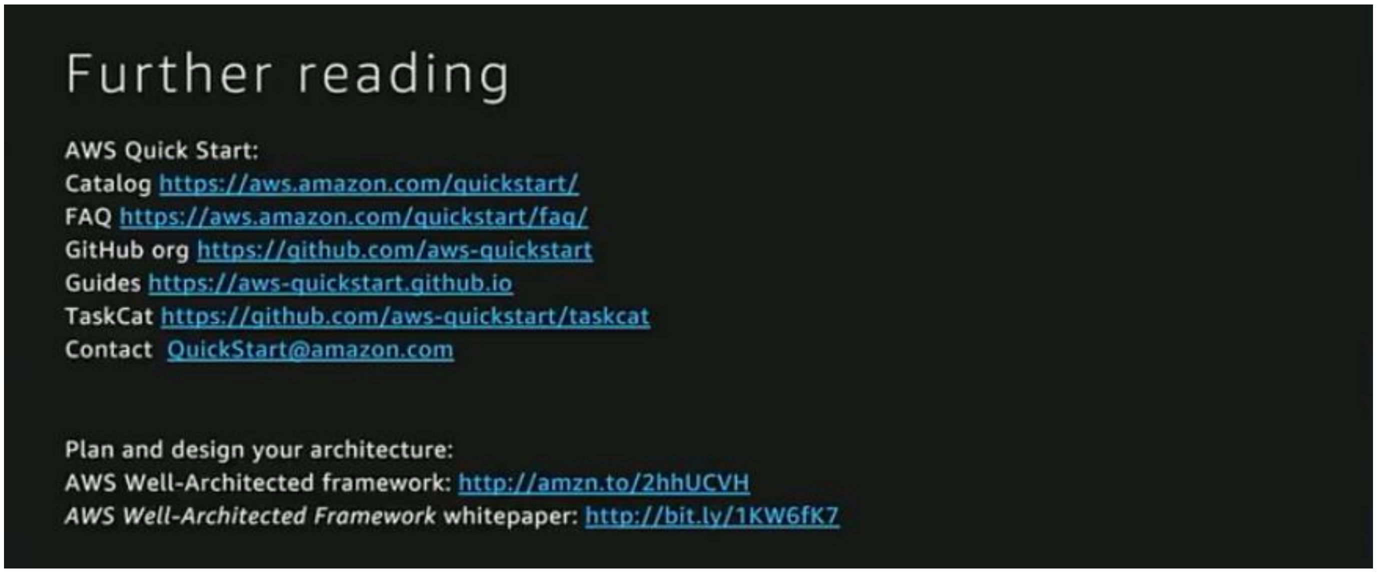
quickstart-aws-vpc
AWS Quick Start Team
★ 29 24

quickstart-linux-bastion
AWS Quick Start Team
Shell ★ 33 15

quickstart-mongodb
AWS Quick Start Team
Shell ★ 16 24

quickstart-git2s3
AWS Quick Start Team
Python ★ 3

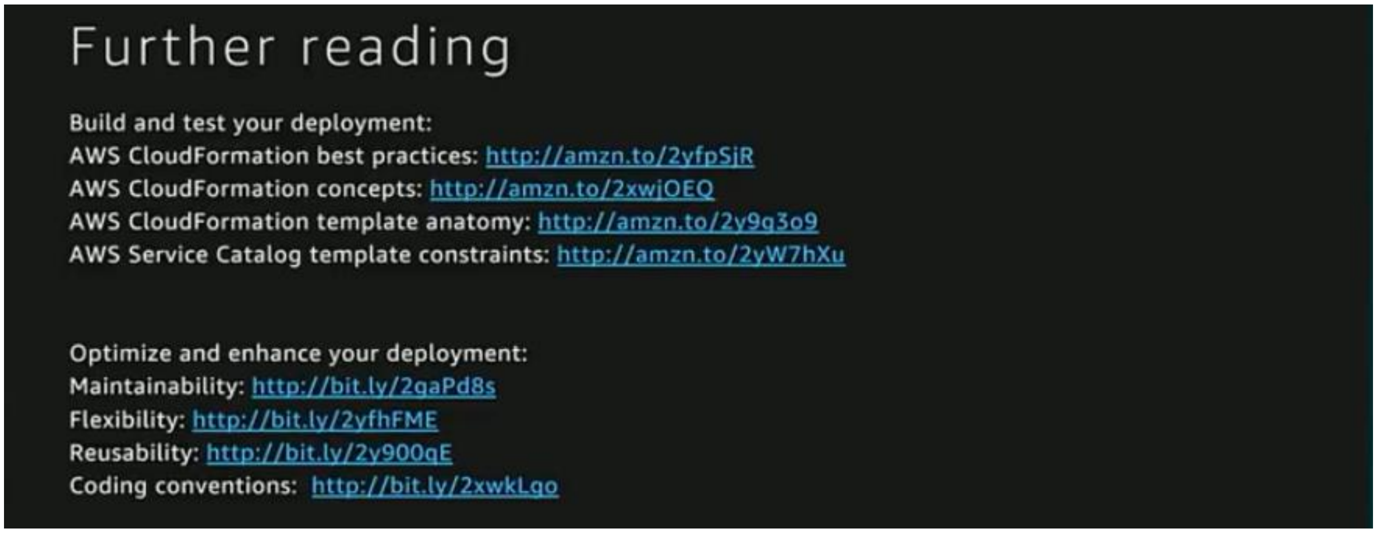
https://github.com/aws-quickstart



Further reading

AWS Quick Start:
Catalog https://aws.amazon.com/quickstart/
FAQ https://aws.amazon.com/quickstart/faq/
GitHub org https://github.com/aws-quickstart
Guides https://aws-quickstart.github.io
TaskCat https://github.com/aws-quickstart/taskcat
Contact QuickStart@amazon.com

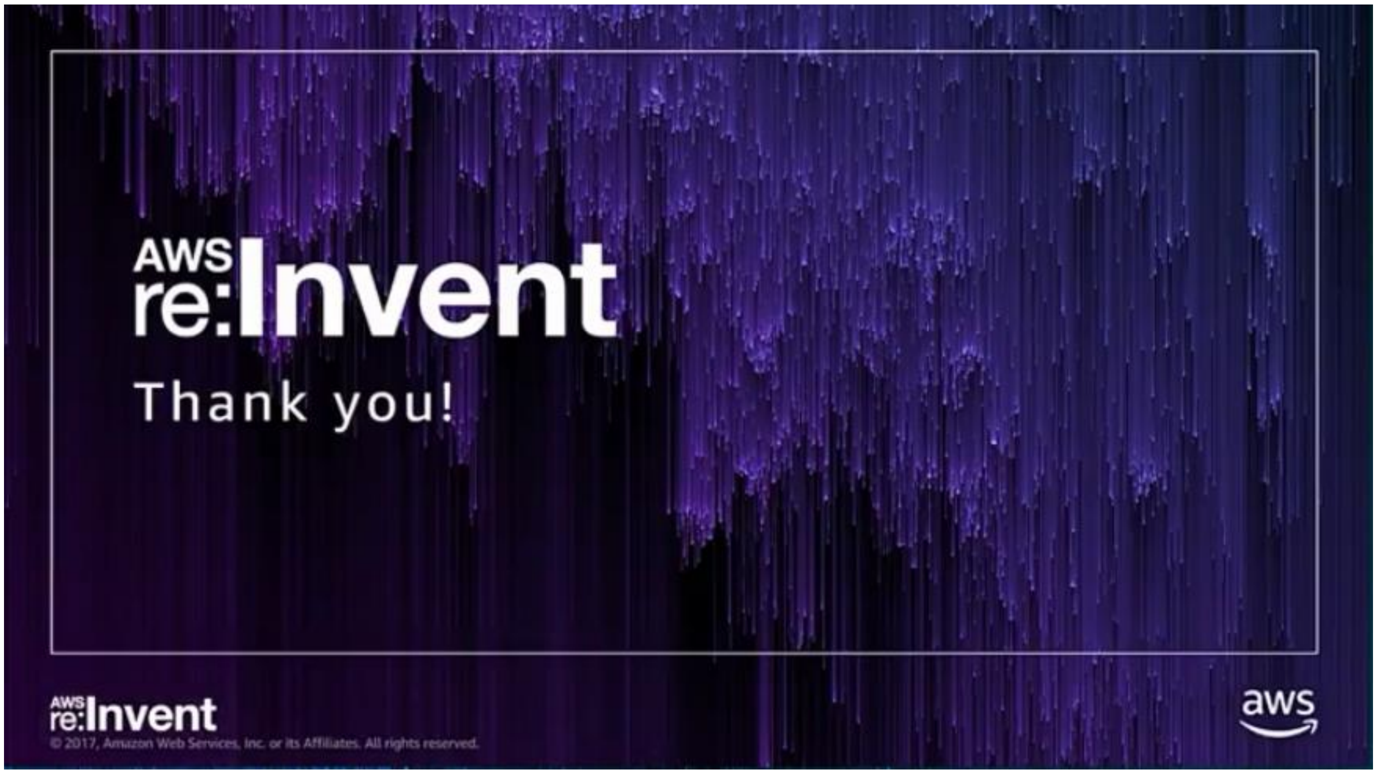
Plan and design your architecture:
AWS Well-Architected framework: http://amzn.to/2hhUCVH
AWS Well-Architected Framework whitepaper: http://bit.ly/1KW6fK7



Further reading

Build and test your deployment:
AWS CloudFormation best practices: http://amzn.to/2yfpSjR
AWS CloudFormation concepts: http://amzn.to/2xwjOEQ
AWS CloudFormation template anatomy: http://amzn.to/2y9q3o9
AWS Service Catalog template constraints: http://amzn.to/2yW7hXu

Optimize and enhance your deployment:
Maintainability: http://bit.ly/2gaPd8s
Flexibility: http://bit.ly/2yfhFME
Reusability: http://bit.ly/2y900qE
Coding conventions: http://bit.ly/2xwkLgo



AWS re:Invent

Thank you!

AWS re:Invent

aws