Deep Dive: AWS Command Line Interface

Thomas Jones, Ecosystem Solutions Architect

amazon web services

Crash Course

Intro to the AWS CLI

FOUNDATION

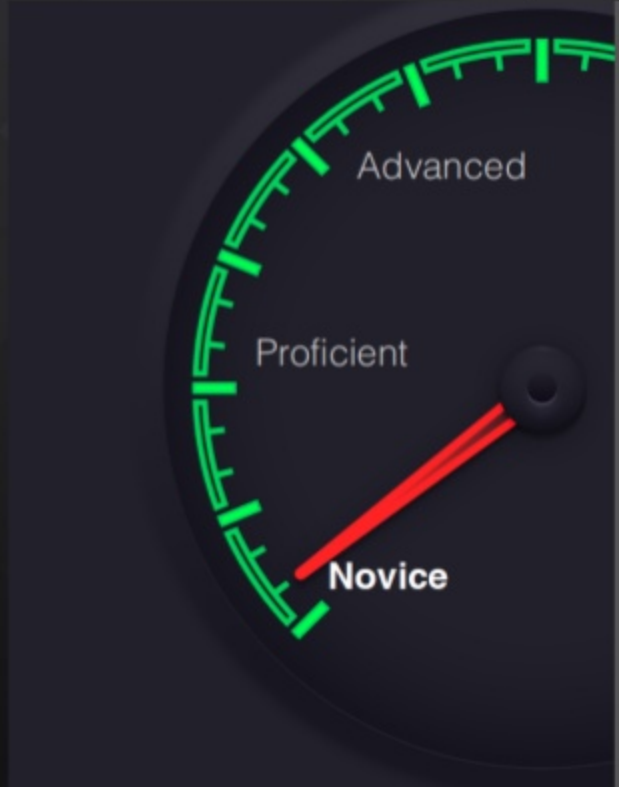
Exploring Key Functionality

ADVANCED SCENARIOS

Looking at Advanced CLI Features

Crash Course

Intro to the AWS CLI



amazon web services

AWS Command Line Interface

Unified tool to manage your AWS services

amazon web services

MSI (Windows)

Bundled (cross platform)

pip (cross platform)

```
aws configure
```

```
$ aws ec2 describe-instances
```

Service (command)

Operation (subcommand)

amazon web services

```
$ aws iam list-access-keys
```

Service (command)

Operation (subcommand)

amazon web services

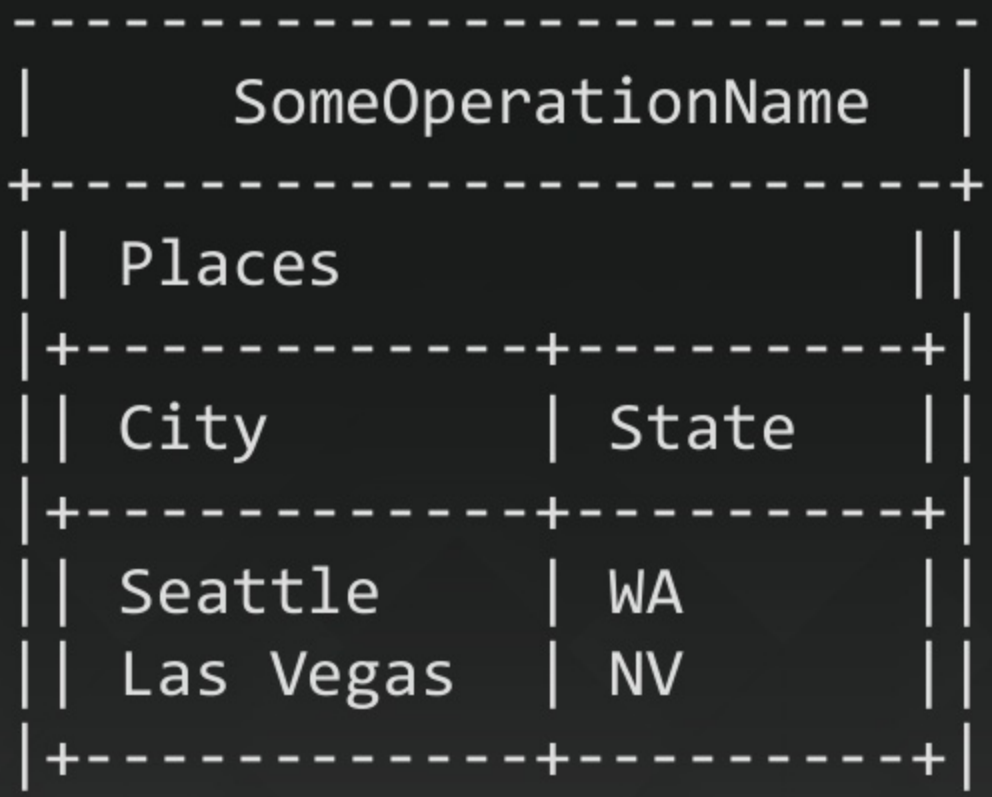
--output JSON

```
{
    "Places": [
        {
            "City": "Seattle",
            "State": "WA"
        },
        {
            "City": ”Las Vegas",
            "State": "NV"
        }
    ]
}
```

--output text

```
PLACES  Seattle    WA
PLACES  Las Vegas  NV
```

--output table



All Outputs

- JSON

```
{
   "Places": [
       {
            "City": "Seattle",
            "State": "WA"
       },
       {
            "City": ”Las Vegas",
            "State": "NV"
       }
   ]
}
```

Text

```
PLACES  Seattle    WA
PLACES  Las Vegas  NV
```

Table

```
-------------------------
|     SomeOperationName  |
+------------------------+
||         Places       ||
|+------------+---------+|
|| City       | State   ||
|+------------+---------+|
|| Seattle    | WA      ||
|| Las Vegas  | NV      ||
|+------------+---------+|
```

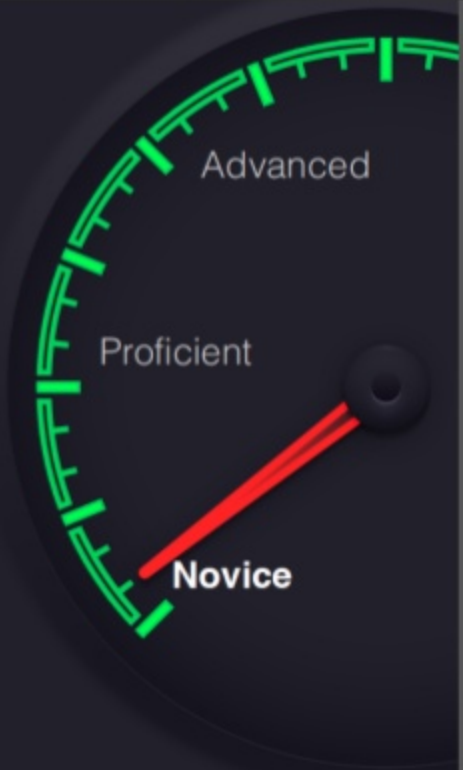
Demo

Basic AWS CLI Usage

amazon web services

Foundation

Exploring Key Functionality



amazon web services

Configuration

amazon web services

```
aws configure
```

aws configure

AWS access key ID [**ABCD]:
AWS secret access key [**********EFGH]:
Default region name [us-west-2]:
Default output format [None]:

```
aws configure <subcommand>
```

aws configure <subcommand>

```
list   - list common configuration sources
get - get the value of a single config var
set - set the value of a single config var
```

```
aws configure get region
```

```
aws configure set profile.prod.region us-west-2
```

A profile is a group of configuration values

```
aws configure --profile prod
```

Configuration Files

~/.aws/credentials

- Supported by all AWS SDKs
- Only contains credentials

~/.aws/config

- Used only by the CLI
- Can contain credentials (but not the default behavior)

~/.aws/credentials

~/.aws/config

```
aws configure set profile.prod.aws_access_key_id foo
```

~/.aws/credentials

~/.aws/config

```
aws configure set profile.prod.aws_access_key_id foo
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
```

~/.aws/config

```
aws configure set profile.prod.aws_secret_access_key bar
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
```

~/.aws/config

```
aws configure set profile.prod.aws_secret_access_key bar
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
aws configure set profile.prod.region uswest2
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
aws configure set profile.prod.region uswest2
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
[profile prod]
region = us-west-2
```

```
aws configure set profile.prod.output text
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
[profile prod]
region = us-west-2
```

```
aws configure set profile.prod.output text
```

~/.aws/credentials

```
[prod]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
[profile prod]
region = us-west-2
output = text
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
     --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
     --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

create-new-user.sh

```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name summit-user
credentials=$(aws iam create-access-key -username summit-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d' ' -f 1)
secret_access_key=$(echo $credentials | cut -d' ' -f 2)
aws configure set profile.summit.aws_access_key_id "$access_key_id"
aws configure set profile.summit.secret_access_key "$secret_access_key"
```

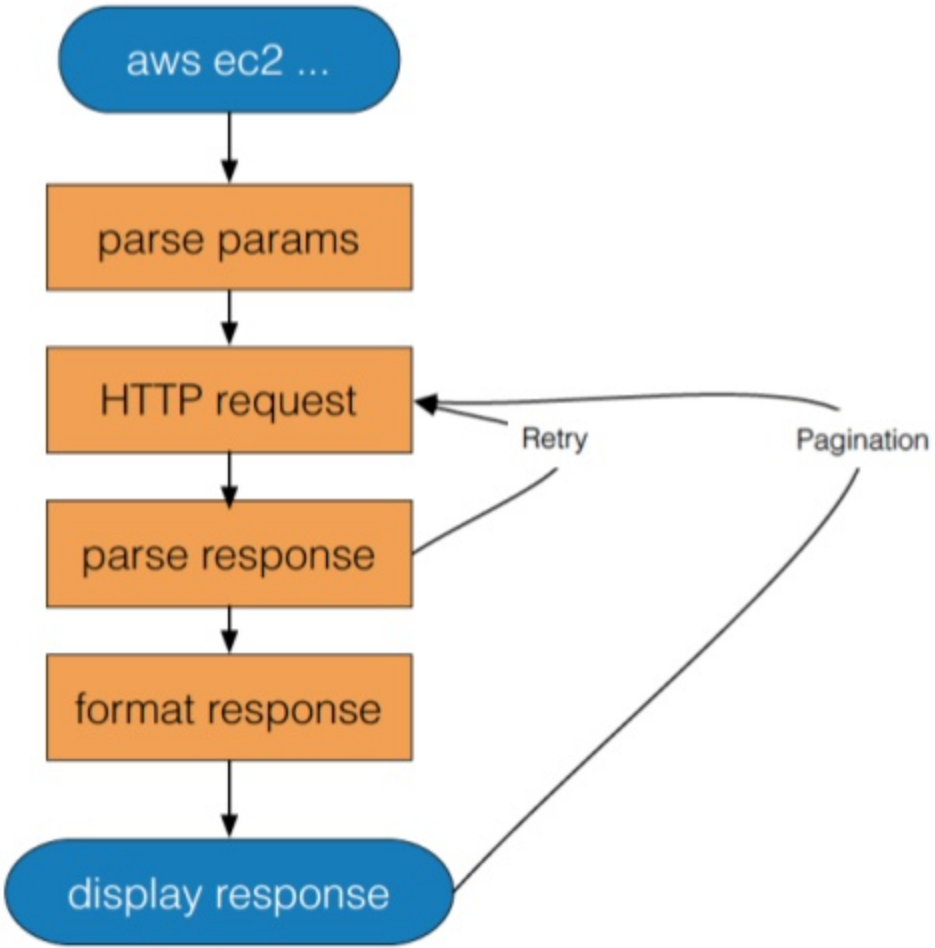
Use the aws configure suite of subcommands

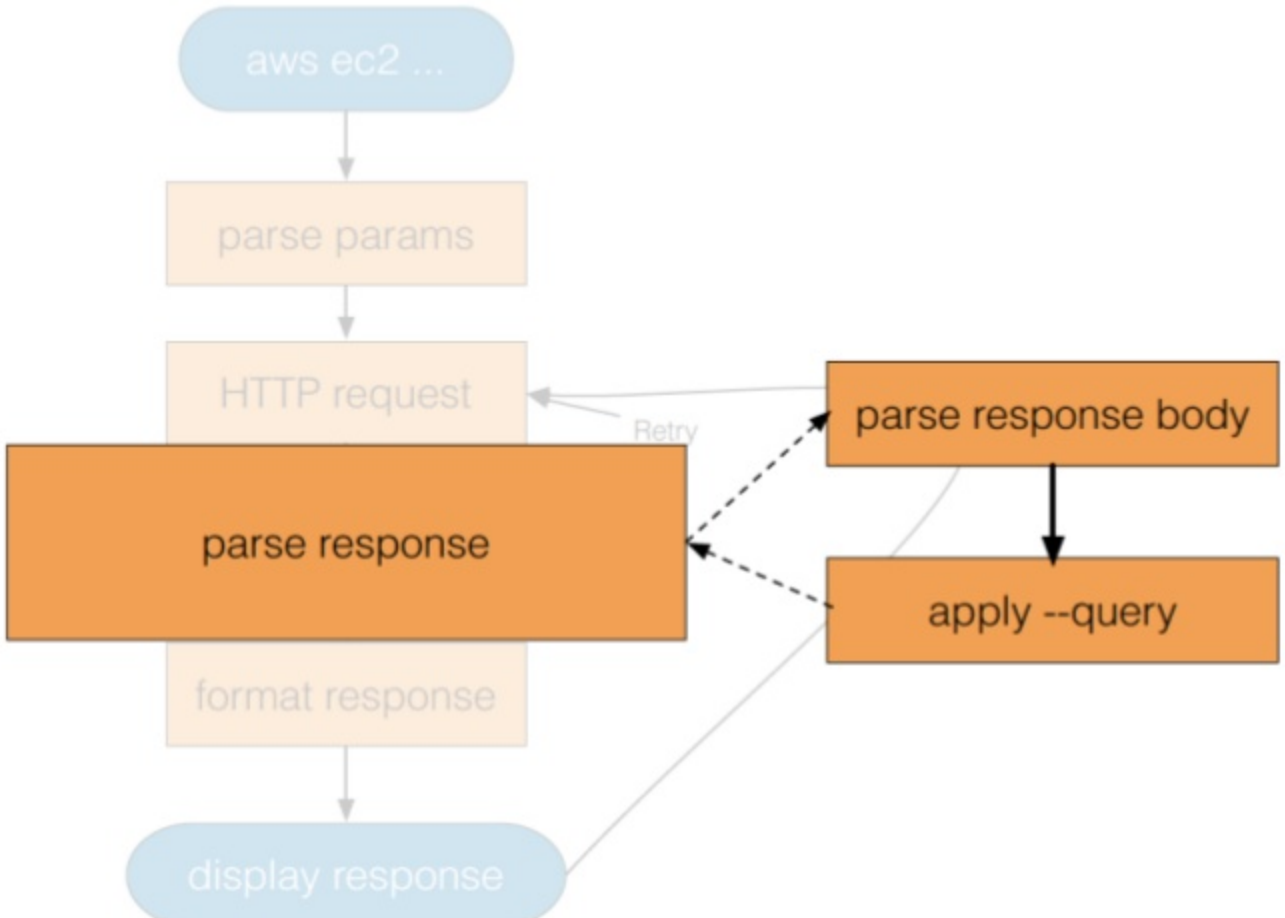
amazon web services

Query

```
-query (string)

A JMESPath query to use in filtering the response data.
```


aws ec2 ...
parse params
HTTP request
Retry
Pagination
parse response
format response
display response


aws ec2 ...
parse params
HTTP request
Retry
parse response
format response
display response
parse response body
apply --query

```
<ListUsersResponse xmlns="...">
  <ListUsersResult>
    <Users>
      <member>
        <UserId>userid</UserId>
        <Path>/</Path>
        <UserName>james</UserName>
        <Arn>arn:aws:iam::..:user/james</Arn>
        <CreateDate>2013-03-09T23:36:32Z</CreateDate>
      </member>
    <Users>
  </ListUsersResult>
<ListUsersResponse>
```

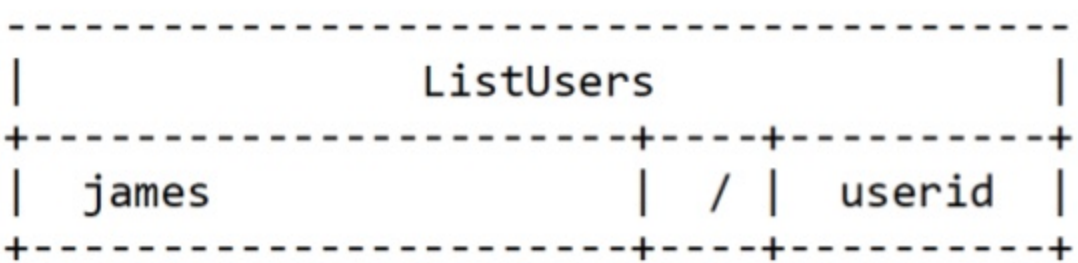
```
{
  "Users": [
    {
      "Arn": "arn:aws:iam::...:user/james",
      "UserId": "userid",
      "CreateDate": "2013-03-09T23:36:32Z",
      "Path": "/",
      "UserName": "james"
    }
  ]
}
```

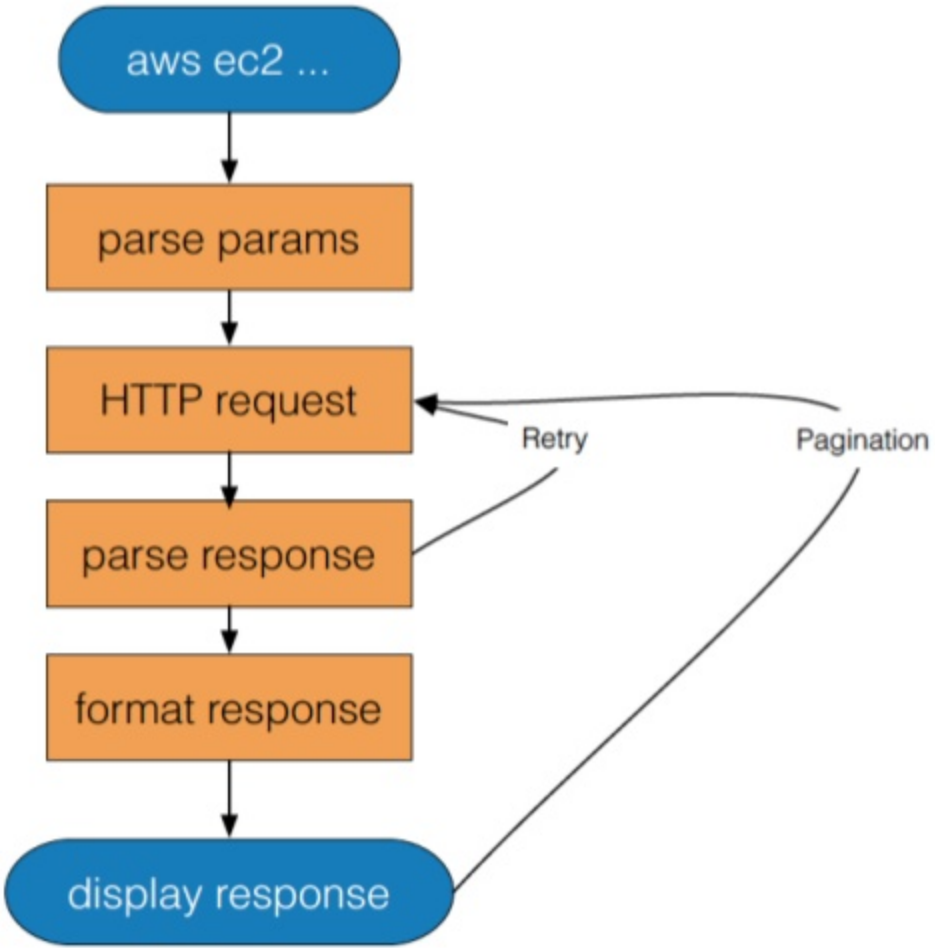
```
--query User[0].[UserName,Path,UserId]
```


```
{
  "Users": [
    {
      "Arn": "arn:aws:iam::...:user/james",
      "UserId": "userid",
      "CreateDate": "2013-03-09T23:36:32Z",
      "Path": "/",
      "UserName": "james"
    }
  ]
}
```

```
--query User[0].[UserName,Path,UserId]
```

```
[
    [
        "james", "/", "id"
    ],
]
```


ListUsers
james
/
userid


aws ec2 ...
parse params
HTTP request
Retry
Pagination
parse response
format response
display response

http://jmespath.org

A Query Language for JSON

Demo

JMESPATH

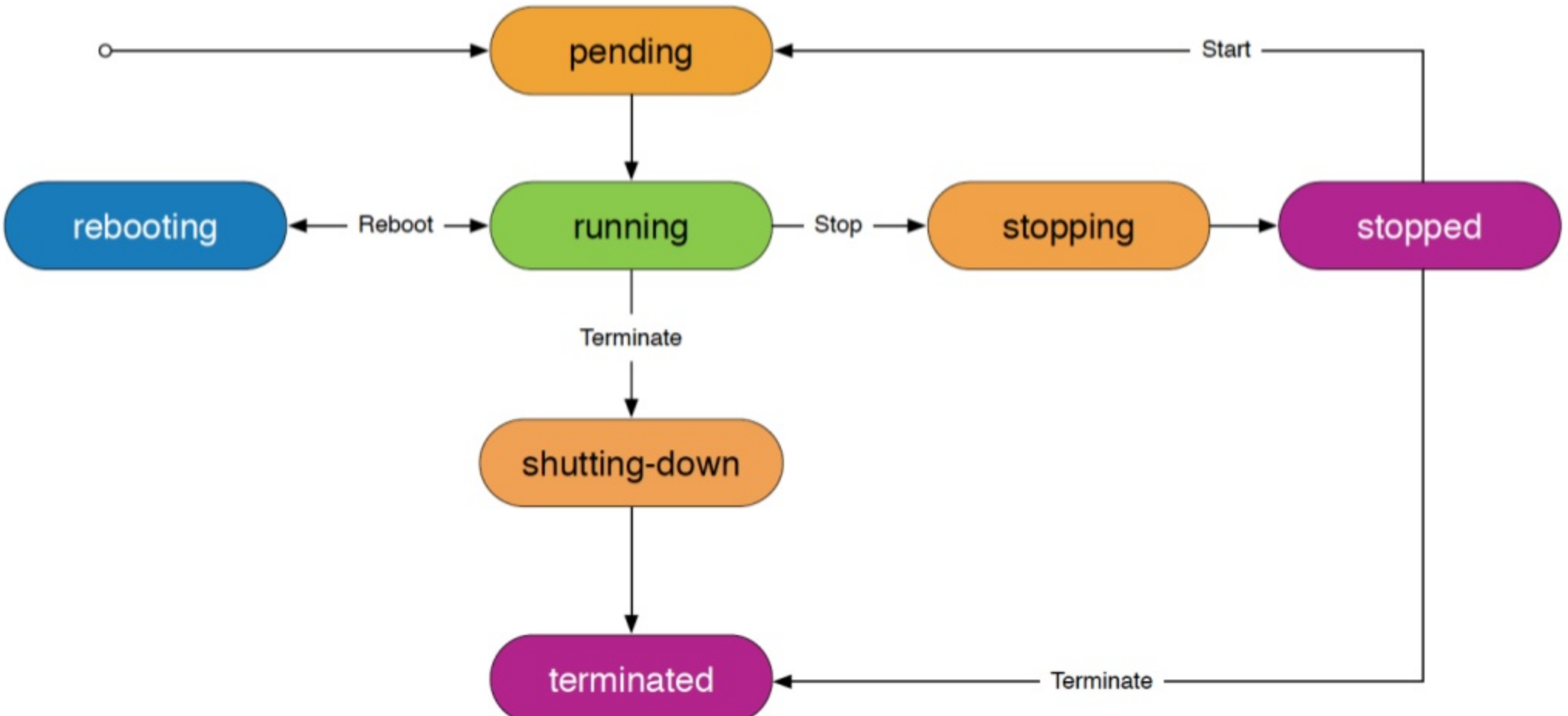
amazon web services

http://jmespath.org

A Query Language for JSON

Waiters

Amazon EC2 Instance State Transitions



ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

ec2-instance-running.sh

```
#!/bin/bash
instance_id=$(aws ec2 run-instances --image-id ami-12345 \
    --query Reservations[].Instances[].instanceId \
    --output text)
instance_state=$(aws ec2 describe-instances --instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
while [ "$instance_state" != "running" ]
do
    sleep 1
    instance_state=$(aws ec2 describe-instances –instance-ids $instance_id \
    --query 'Reservations[].Instances[].State.Name')
done
```

- No timeouts
- Failure states
- Hand-written code

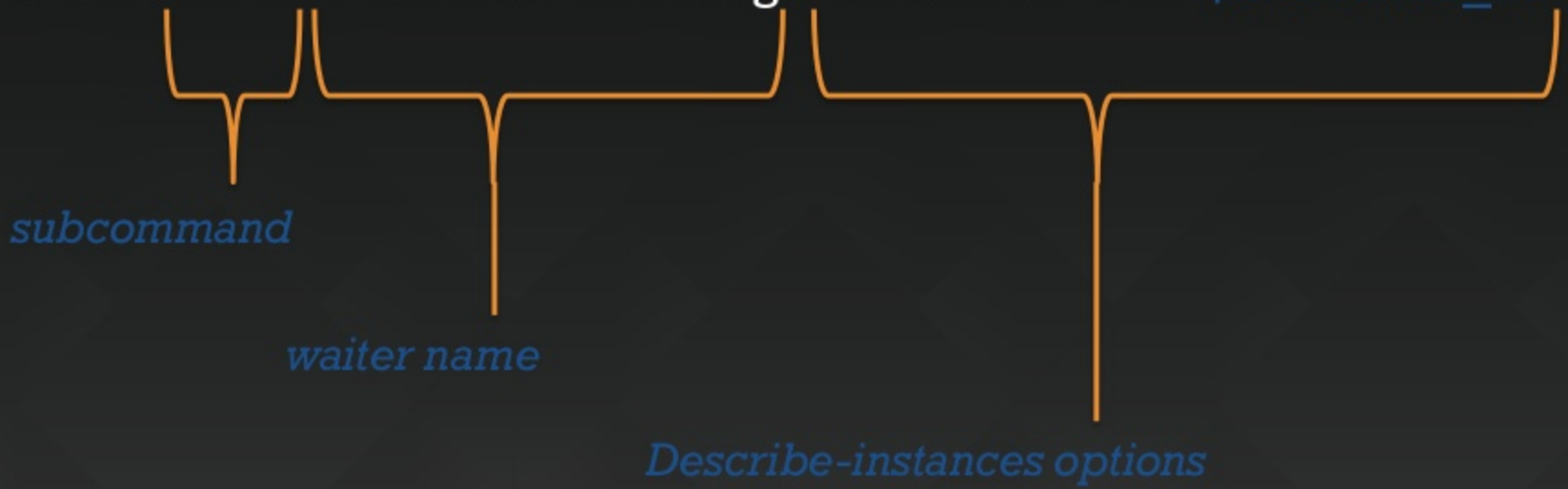
amazon web services

ec2-waiters.sh

```
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
aws ec2 wait instance-running –instance-ids $instance_id
```

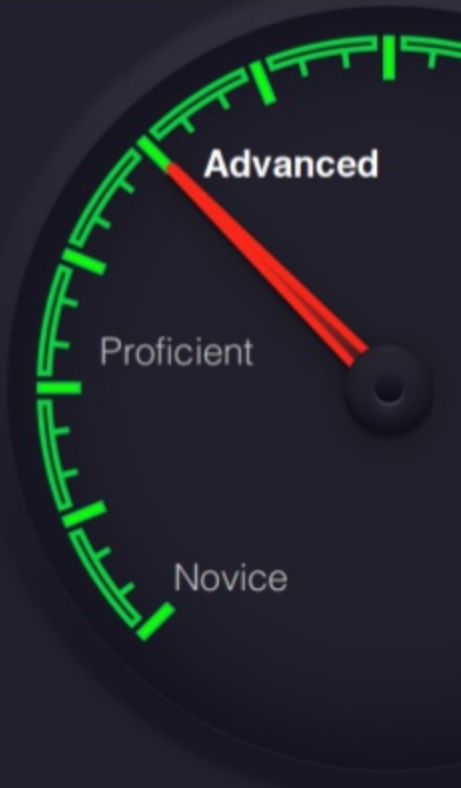
ec2-waiters.sh

```
instance_id=$(aws ec2 run-instances –image-id ami-12345 \
    --query Reservations[].Instances[].InstanceId \
    --output text)
aws ec2 wait instance-running –instance-ids $instance_id
```



Advanced Scenarios

Looking at advanced AWS CLI features



amazon web services

Templates

amazon web services

The AWS CLI is data driven

```
"RunInstancesRequest":{
  "type":"structure",
  "required":[
    "ImageId",
    "MinCount",
    "MaxCount"
  ],
  "members":{
    "ImageId":{"shape":"String"},
    "MinCount":{"shape":"Integer"},
    "MaxCount":{"shape":"Integer"},
    "KeyName":{"shape":"String"},
    "SecurityGroups":{
      "shape":"SecurityGroupStringList",
      "locationName":"SecurityGroup"
    },
```

```
"RunInstancesRequest":{
  "type":"structure",
  "required":[
    "ImageId",
    "MinCount",
    "MaxCount"
  ],
  "members":{
    "ImageId":{"shape":"String"},
    "MinCount":{"shape":"Integer"},
    "MaxCount":{"shape":"Integer"},
    "KeyName":{"shape":"String"},
    "SecurityGroups":{
      "shape":"SecurityGroupStringList",
      "locationName":"SecurityGroup"
    },
```

```
"RunInstancesRequest":{
  "type":"structure",
  "required":[
    "ImageId",
    "MinCount",
    "MaxCount"
  ],
  "members":{
    "ImageId":{"shape":"String"},
    "MinCount":{"shape":"Integer"},
    "MaxCount":{"shape":"Integer"},
    "KeyName":{"shape":"String"},
    "SecurityGroups":{
      "shape":"SecurityGroupStringList",
      "locationName":"SecurityGroup"
    },
```

arguments.json

```
{
"ImageId": "ami-12345",
"MinCount": 1,
"MaxCount": 1,
"KeyName": "id_rsa",
"SecurityGroups": ["SSH", "web"],
}
```

```
aws ec2 run-instances –cli-input-json file://arguments.json
```

What else can we do?

```
aws ec2 run-instances –generate-cli-skeleton
```

Demo

Creating and using JSON templates

amazon web services

Credential Providers

amazon web services

Credential Providers

```
export AWS_ACCESS_KEY_ID=...
```

~/.aws/credentials

CredentialLocator

~/.aws/config

Amazon EC2
Instance Metadata

Credential Providers

```
export AWS_ACCESS_KEY_ID=...
```

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

CredentialLocator

Credential Providers

```
export AWS_ACCESS_KEY_ID=...
```

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

CredentialLocator

Credential Providers

```
export AWS_ACCESS_KEY_ID=...
```

~/.aws/credentials

CredentialLocator

~/.aws/config

Amazon EC2
Instance Metadata

Credential Providers

CredentialLocator

```
export AWS_ACCESS_KEY_ID=...
```

~/.aws/credentials

~/.aws/config

AssumeRole

Amazon EC2
Instance Metadata

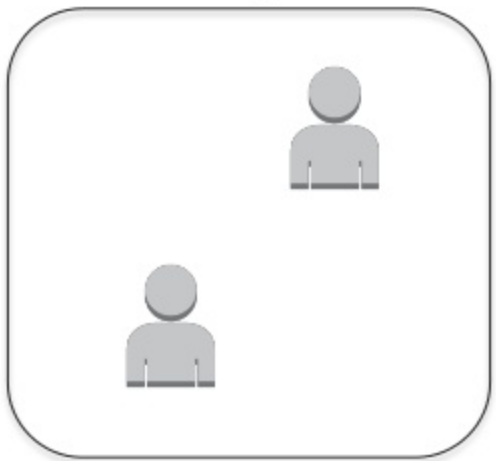
Delegate access to AWS resources using

AWS Identity and Access Management (IAM) roles

IAM Roles

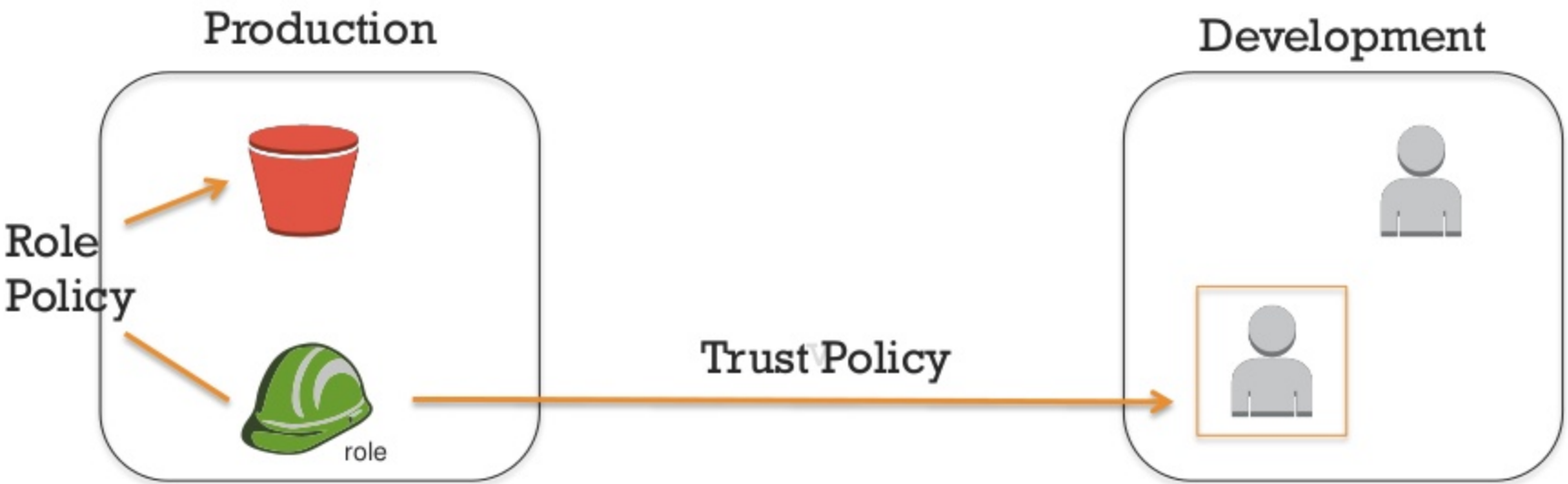
Production

Development

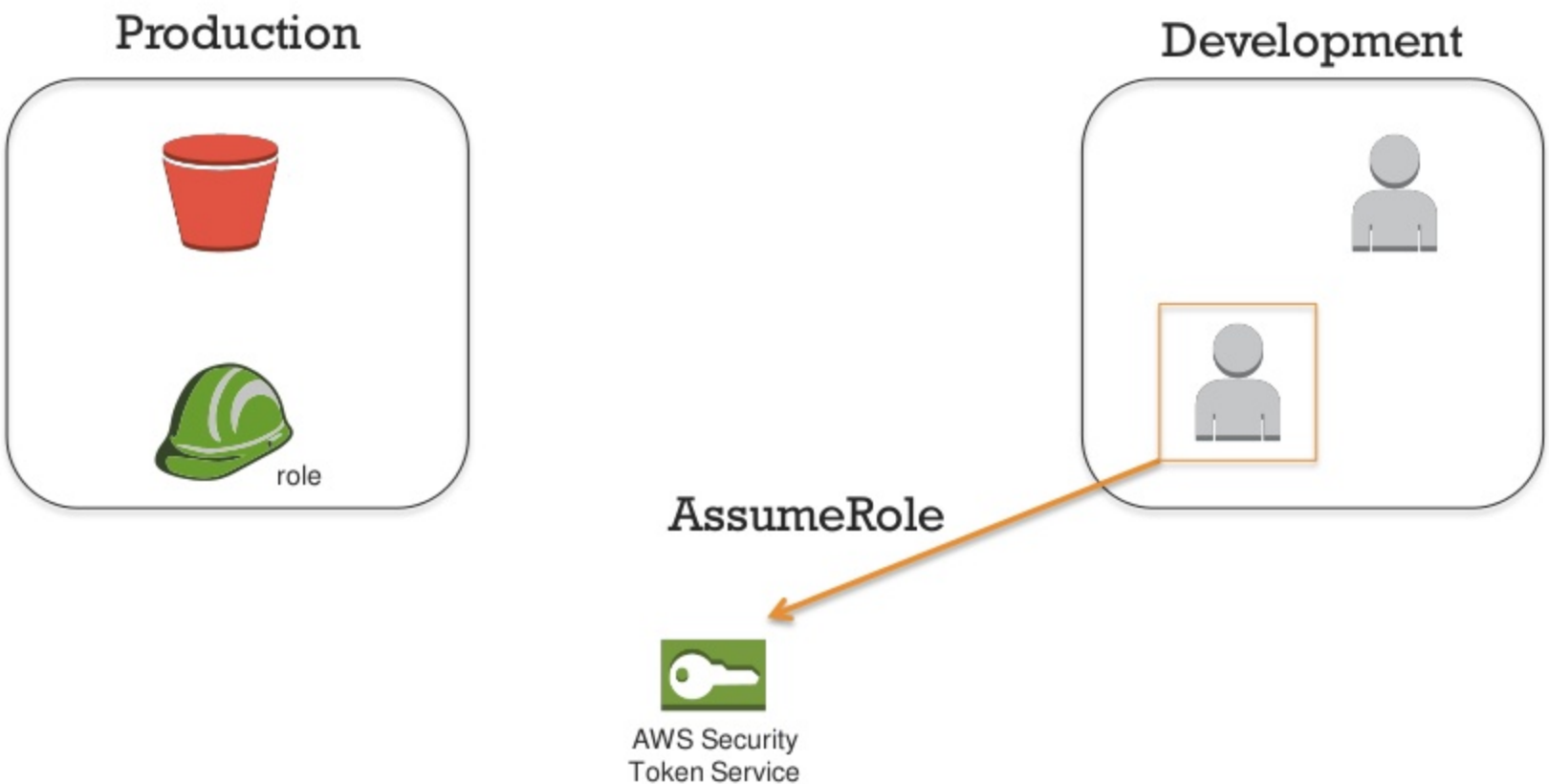


IAM Roles

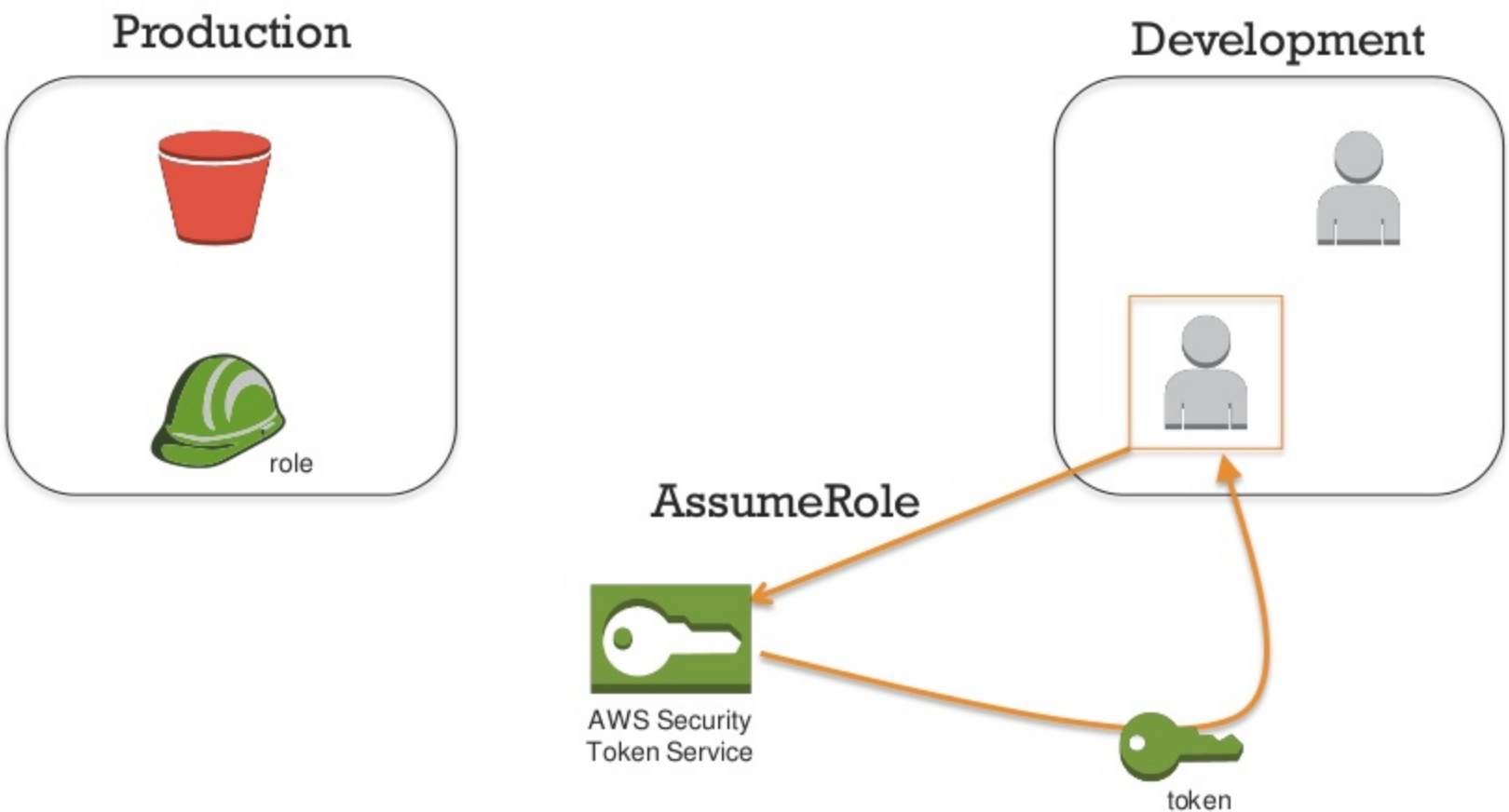
IAM Roles



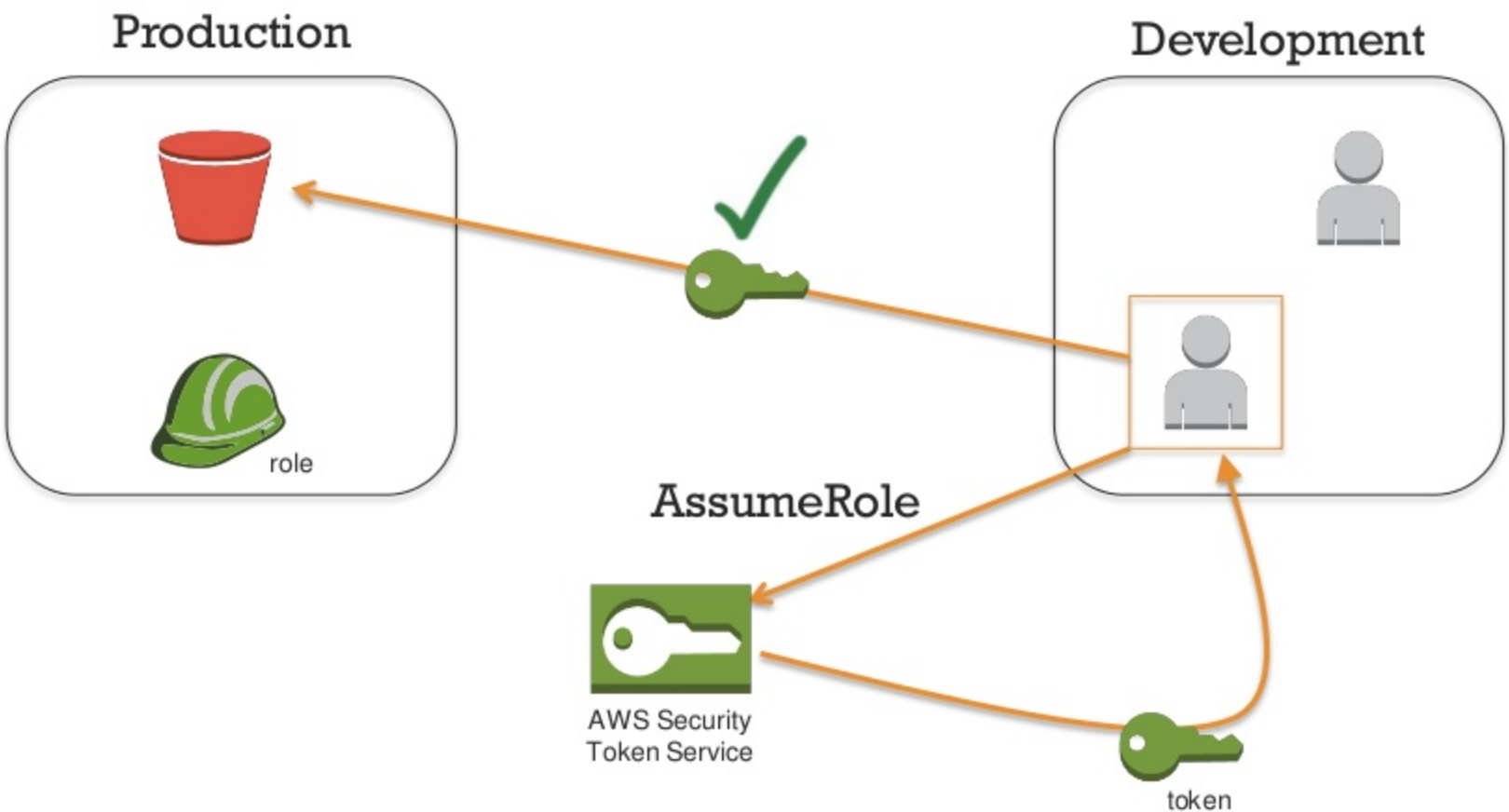
IAM Roles


Production
role
Development
AssumeRole
AWS Security
Token Service

IAM Roles


Production
role
Development
AssumeRole
AWS Security
Token Service
token

IAM Roles


Production
Development
role
AssumeRole
AWS Security
Token Service
token

configure-role.sh

```
aws configure set profile.prodrole.role_arn arn:aws:iam…

aws configure set profile.prodrole.source_profile dev
```

~/.aws/credentials

```
[dev]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

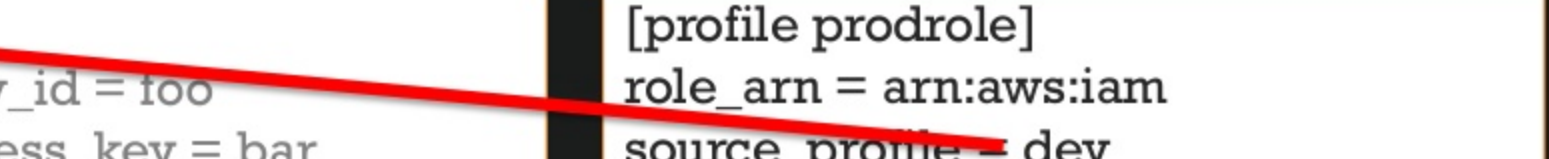
```
[profile prodrole]
role_arn = arn:aws:iam
source_profile = dev
```

~/.aws/credentials

```
[dev]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

```
[profile prodrole]
role_arn = arn:aws:iam
source_profile = dev
```



Demo

Using roles with the AWS CLI

amazon web services

Amazon S3 Streaming

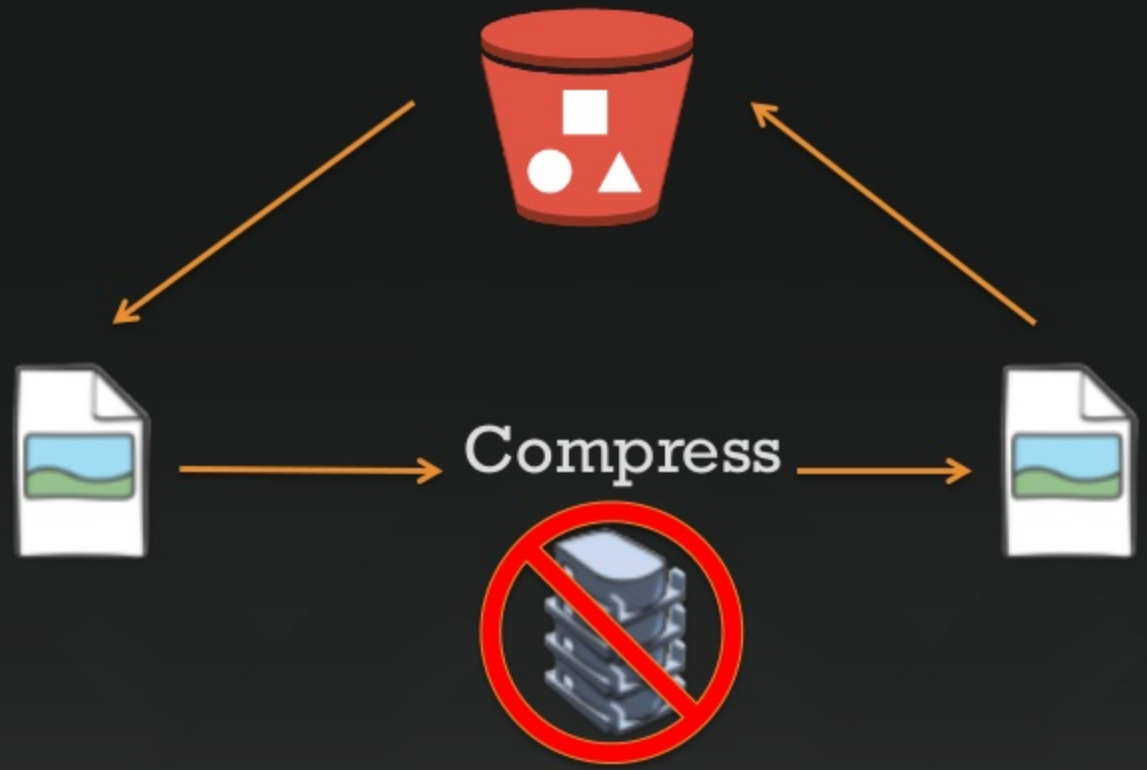
amazon web services

```
aws s3 cp
```

We want to avoid disk

```
aws s3 cp - s3://bucket/key
```

```
aws s3 cp s3://bucket/key -
```



```
aws s3 cp s3://bucket/key - | \
bzip2 -best | \
aws s3 cp - s3://bucket/key.bz2
```

Summary

amazon web services

Wrapping Up

- Configuration
- Waiters
- Query
- Templates
- Credential Providers
- Amazon S3 Streaming

For More Information

- https://github.com/aws/aws-cli
- http://docs.aws.amazon.com/cli/latest/userguide/
- https://forums.aws.amazon.com/forum.jspa?forumID=150
- http://docs.aws.amazon.com/cli/latest/reference/
- http://jmespath.org/

Thank you!

tpjones@amazon.com

amazon web services