Masterclass
Advanced Usage of the AWS CLI

Danilo Poccia

AWS Technical Evangelist

@danilop

danilop

Masterclass

A technical deep dive beyond the basics
Help educate you on how to get the best from AWS technologies
Show you how things work and how to get things done
Broaden your knowledge in ~45 minutes

Crash Course

Intro to the AWS CLI

Foundation

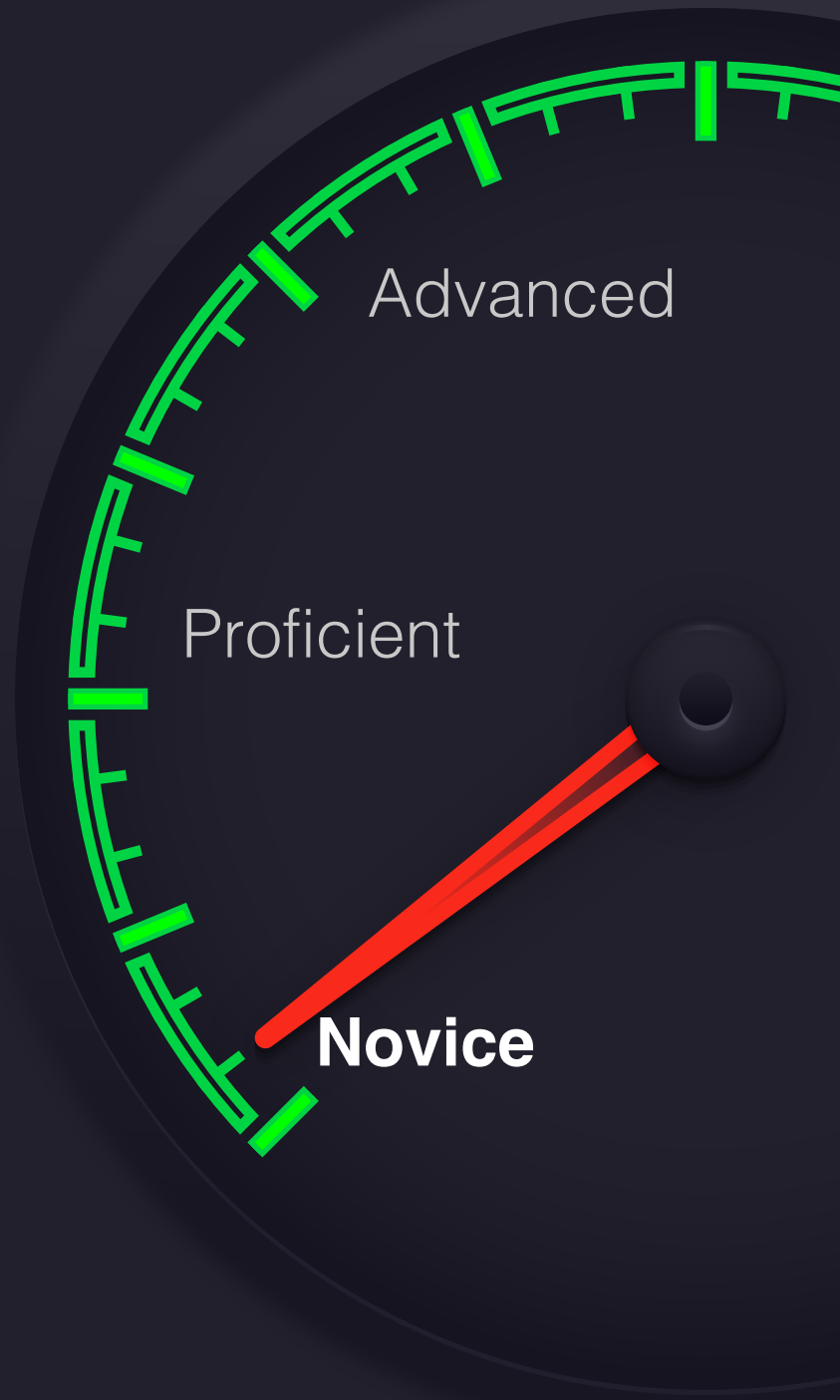
Exploring Key Functionality

Advanced Scenarios

Looking at Advanced CLI Features

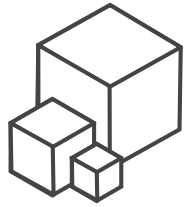
Crash Course

Intro to the AWS CLI



AWS Command Line Interface

Unified tool to manage your AWS services



MSI (Windows)

Bundled (cross platform)

pip (cross platform)

```
aws configure
```

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

service (command)

operation (subcommand)


$ aws iam list-access-keys
service (command)
operation (subcommand)

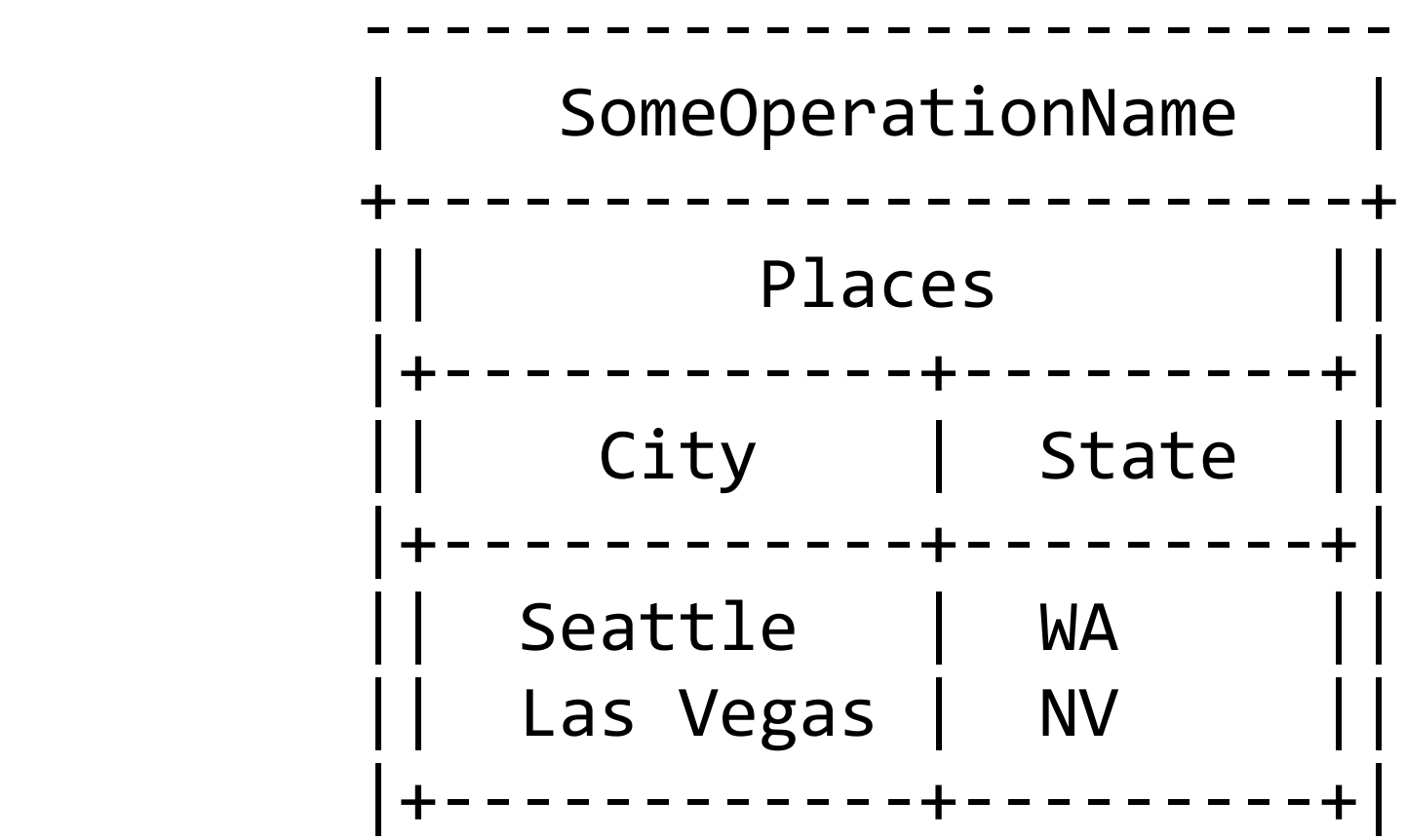
--output json

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

--output text

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

--output table



JSON

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

Text

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

Table

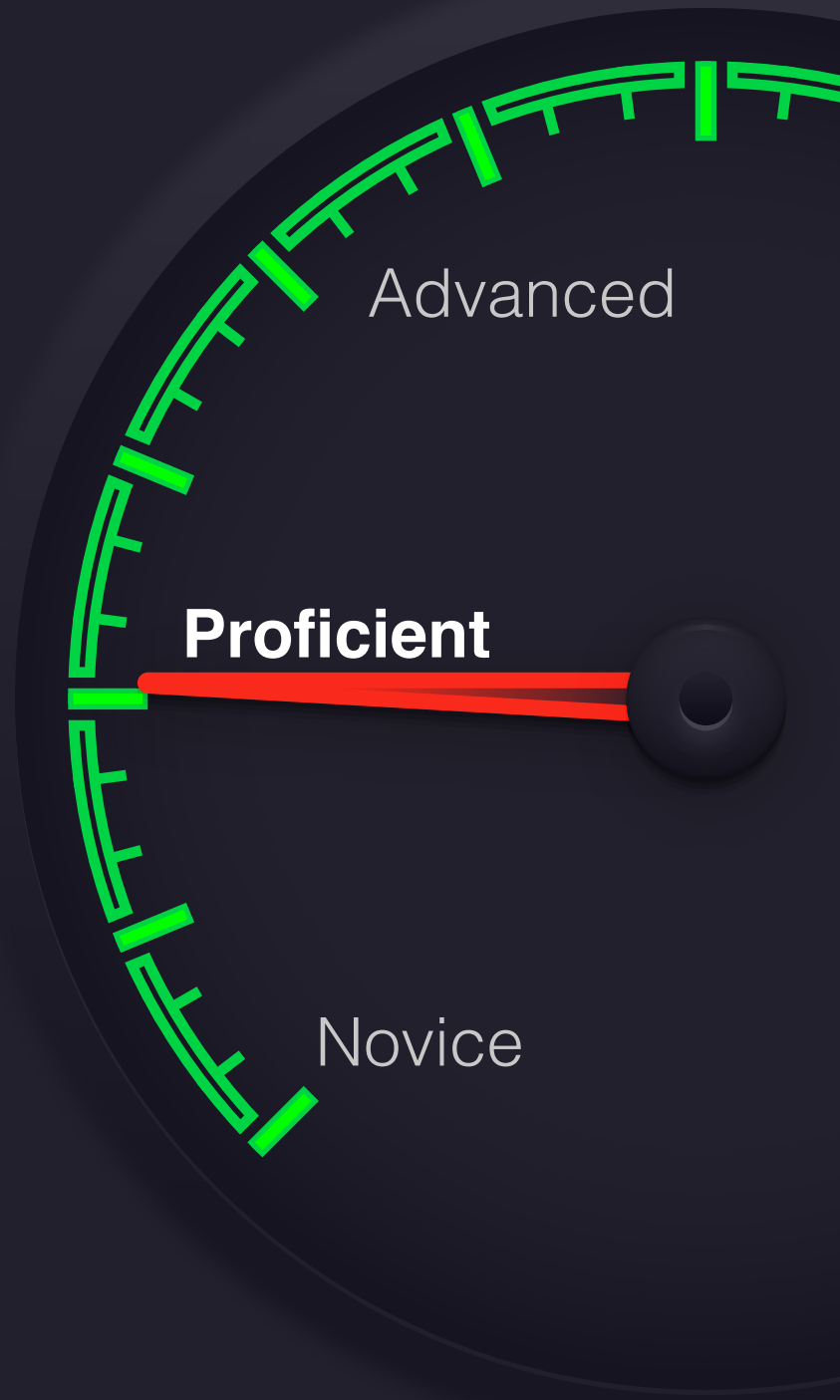
```
-------------------------
|    SomeOperationName   |
+------------------------+
||        Places        ||
|+------------+---------+|
||    City    |  State  ||
|+------------+---------+|
||  Seattle   |  WA     ||
||  Las Vegas |  NV     ||
|+------------+---------+|
```

Basic AWS CLI Usage

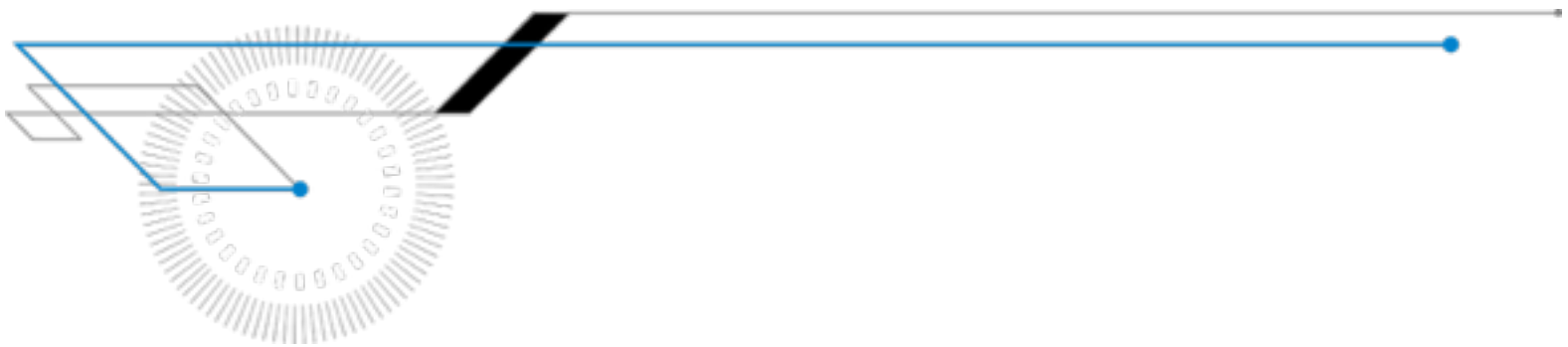
DEMO

Foundation

Exploring Key Functionality



Configuration



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 us-west-2

~/.aws/credentials

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

~/.aws/config

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

~/.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
```

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

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

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

create-new-user.sh

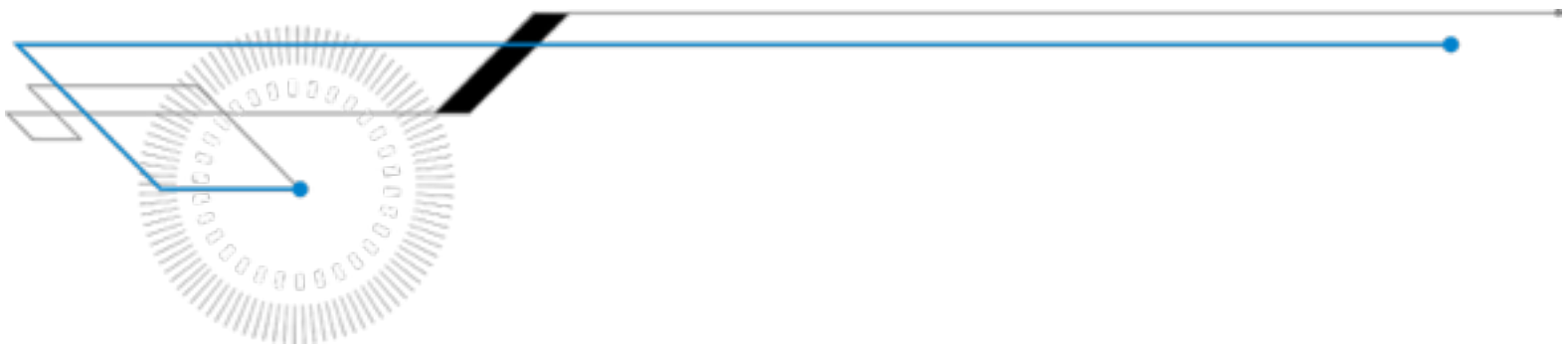
```
#!/bin/bash
# Create a new user and create a new profile.
aws iam create-user --user-name masterclass-user
credentials=$(aws iam create-access-key --user-name masterclass-user \
    --query 'AccessKey.[AccessKeyId,SecretAccessKey]' \
    --output text)
access_key_id=$(echo $credentials | cut -d'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

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

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

Use the aws configure suite of subcommands

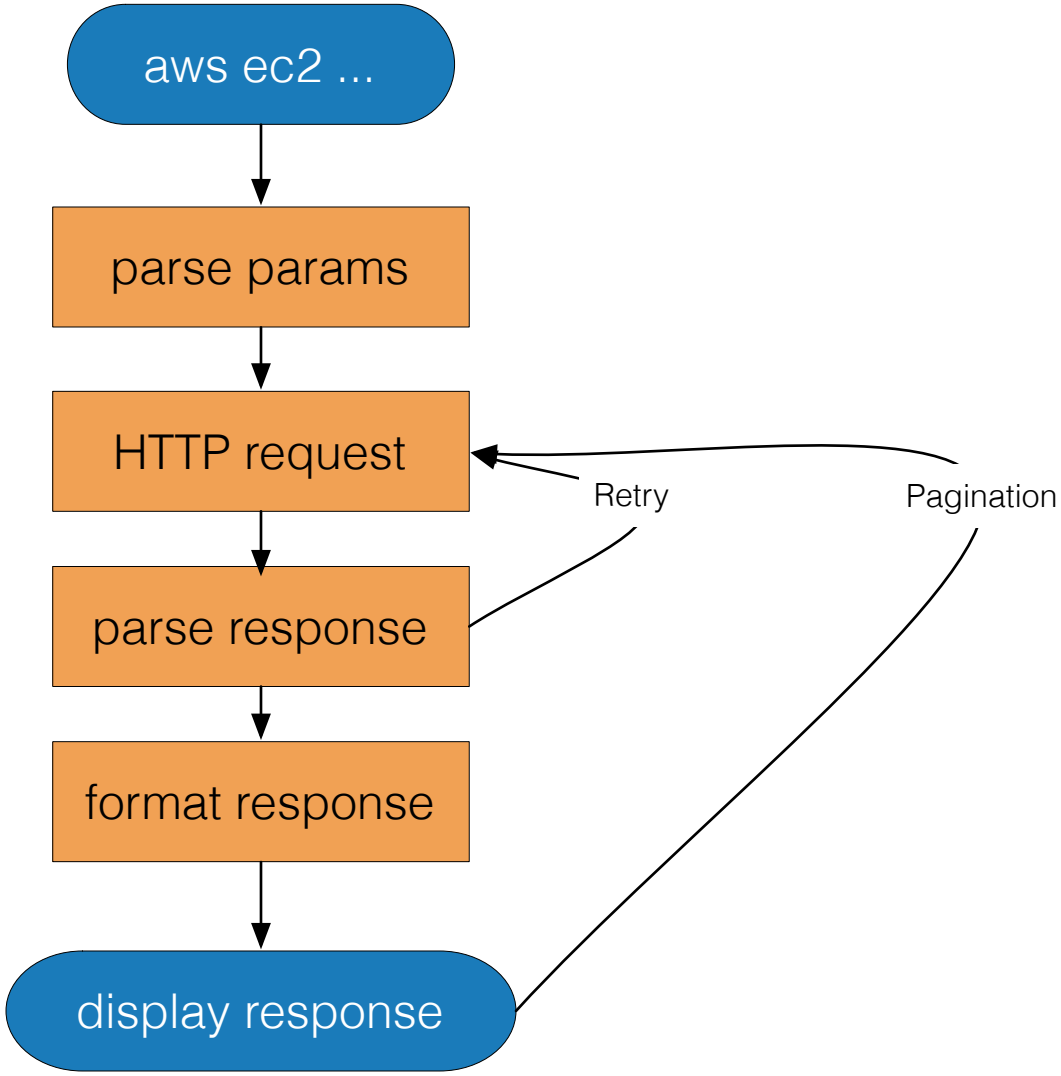
Query



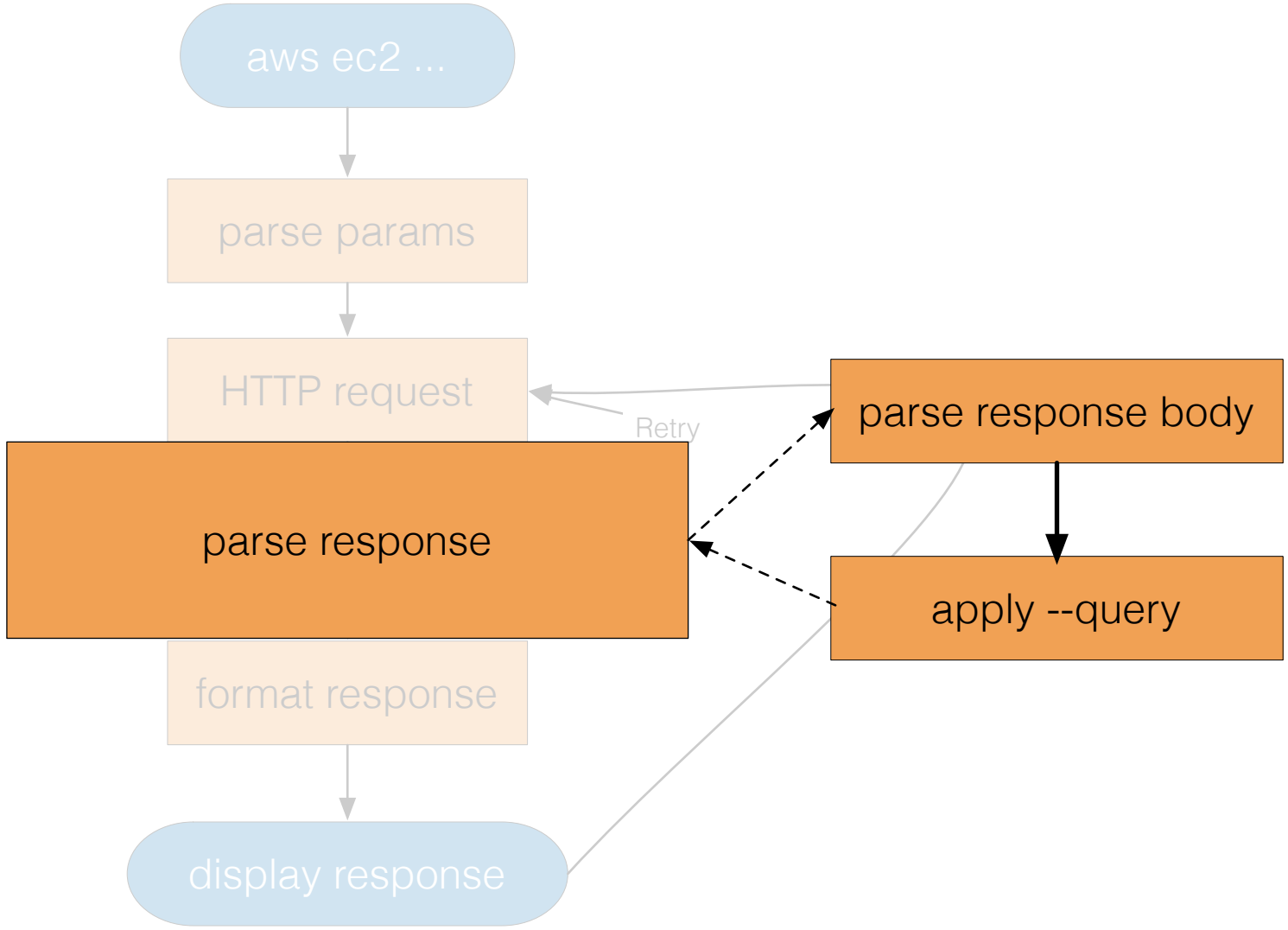
--query (string)

A JMESPath query to use in filtering the response data.

Implementation Details



Implementation Details



Implementation Details

```
<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>
```

Implementation Details

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

Implementation Details

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

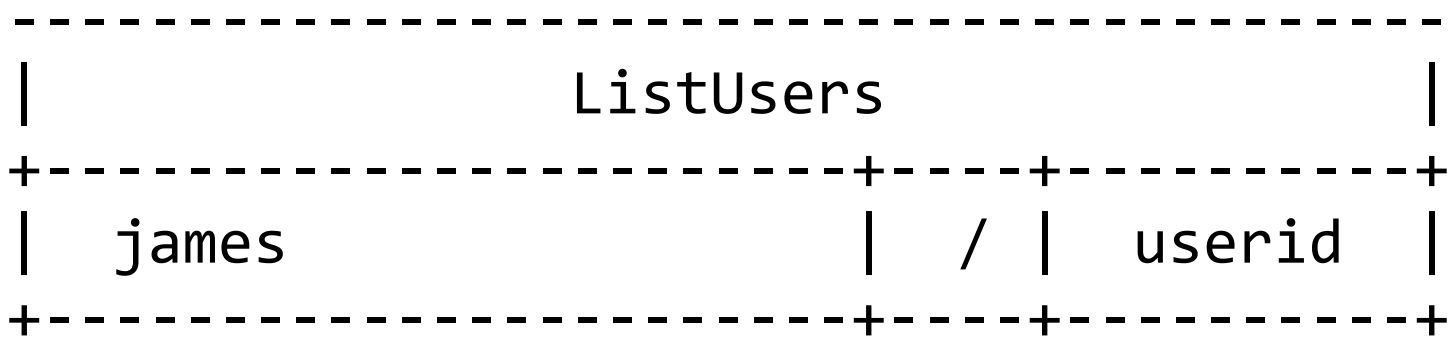
```
{
  "Users": [
    {
      "Arn": "arn:aws:iam::...:user/james",
      "UserId": "userid",
      "CreateDate": "2013-03-09T23:36:32Z",
      "Path": "/",
      "UserName": "james"
    }
  ]
}
```

Implementation Details

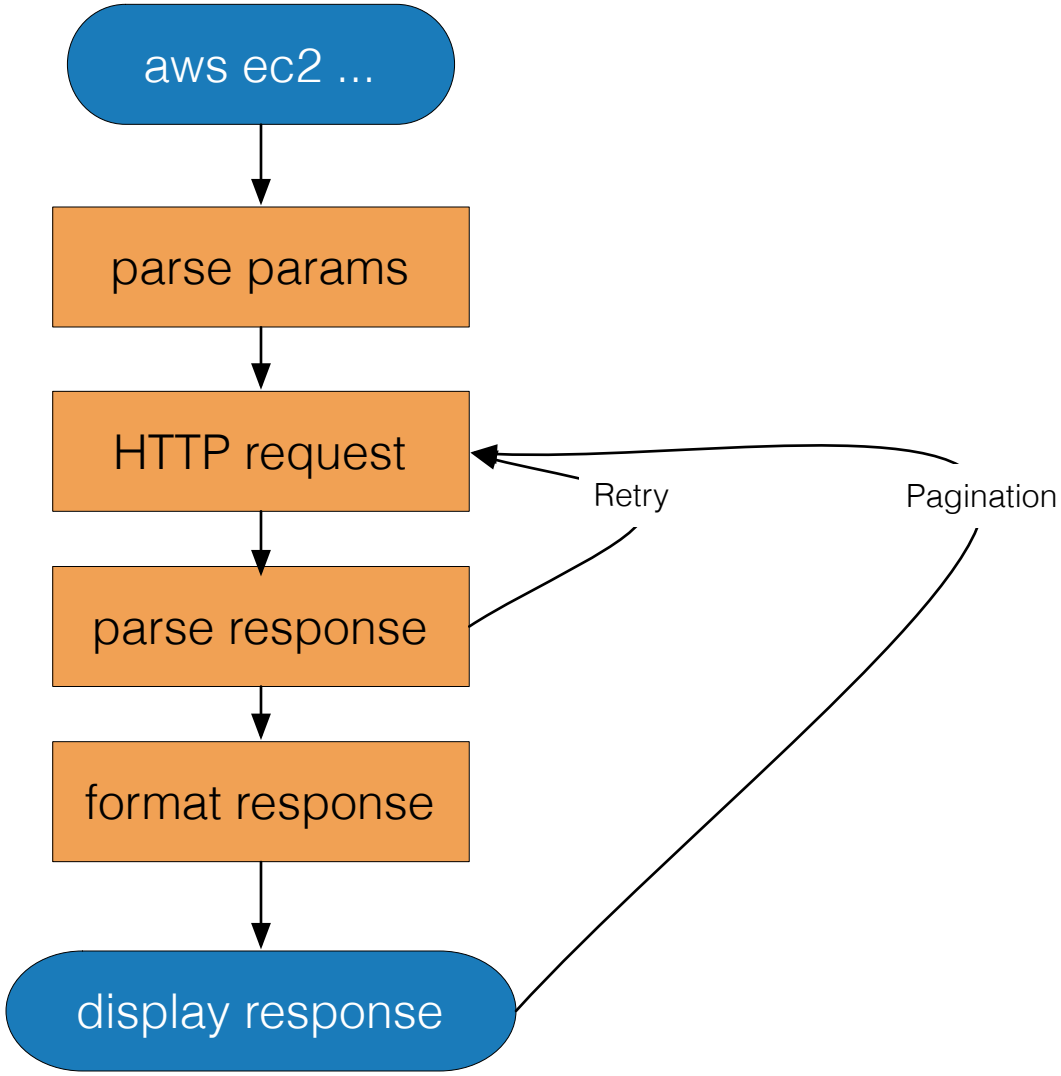
```
--query Users[0].[UserName,Path,UserId]
```

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

Implementation Details



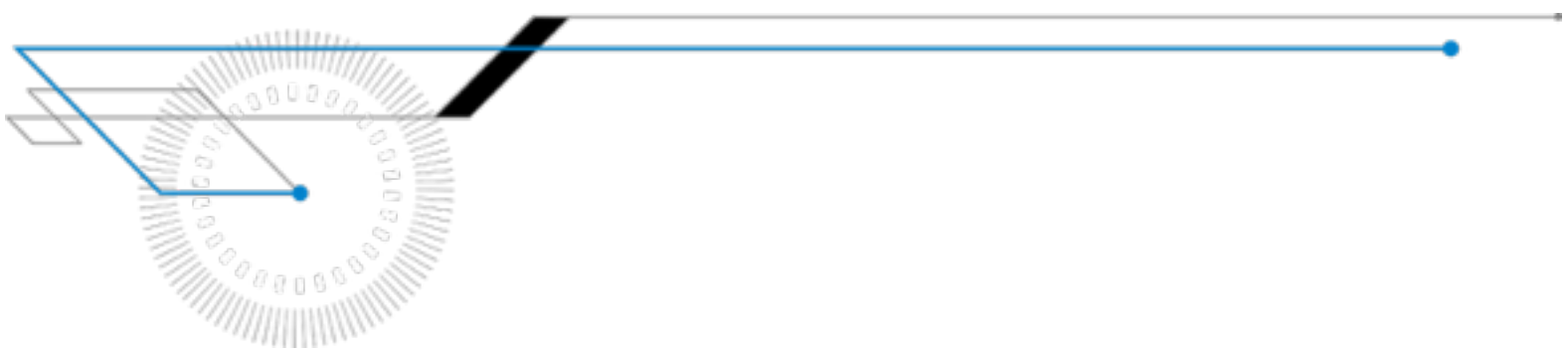
Implementation Details



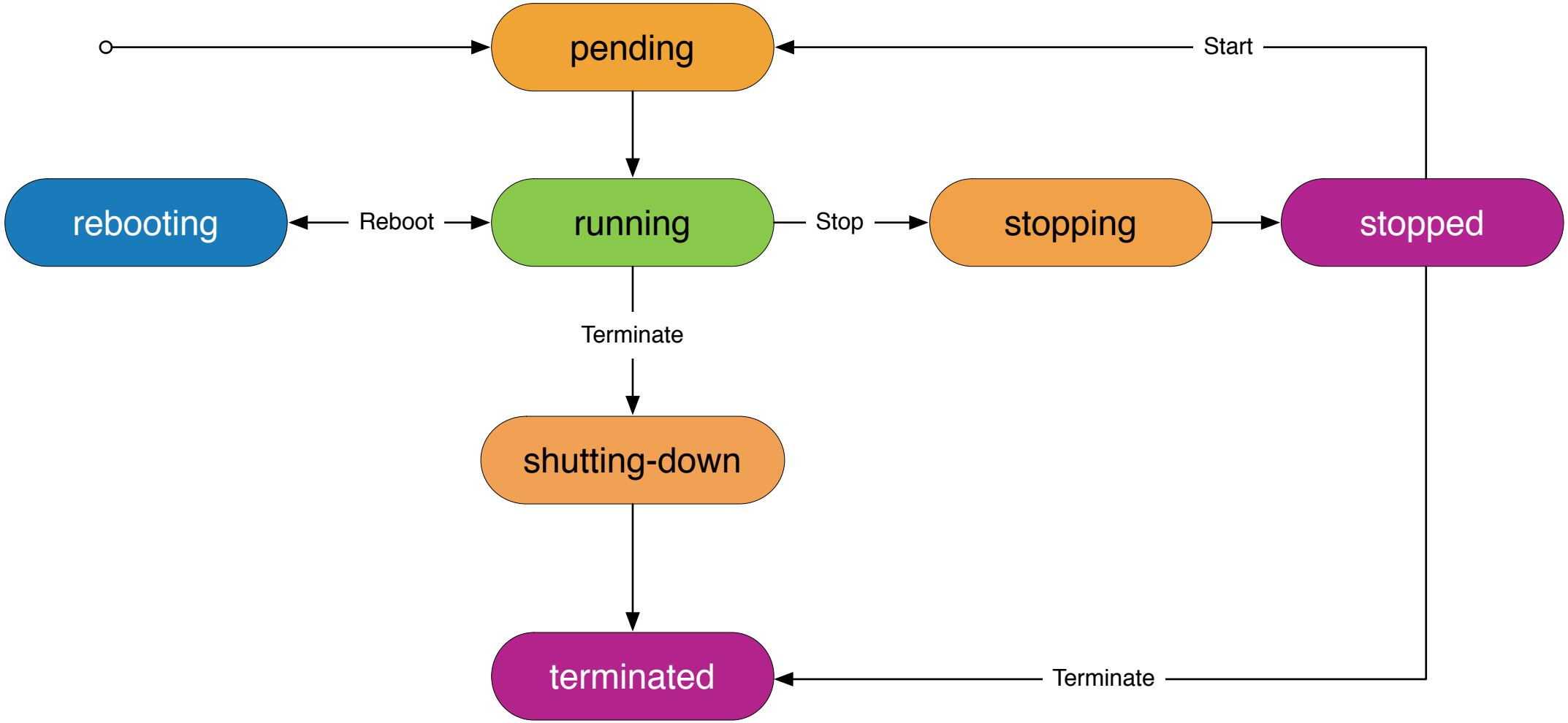
http://jmespath.org

A Query Language For JSON

Waiters



Amazon EC2 Instance State Transitions



```
#!/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
```

```
#!/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
```

```
#!/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
```

```
#!/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
```

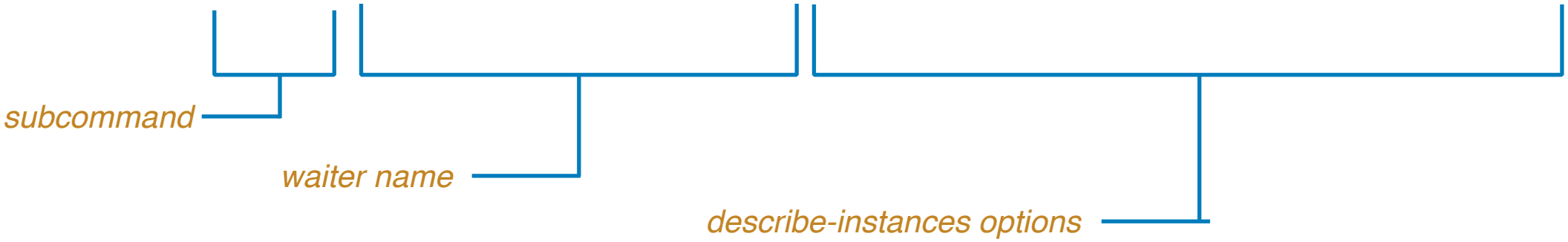
ec2-instance-running.sh

```
#!/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

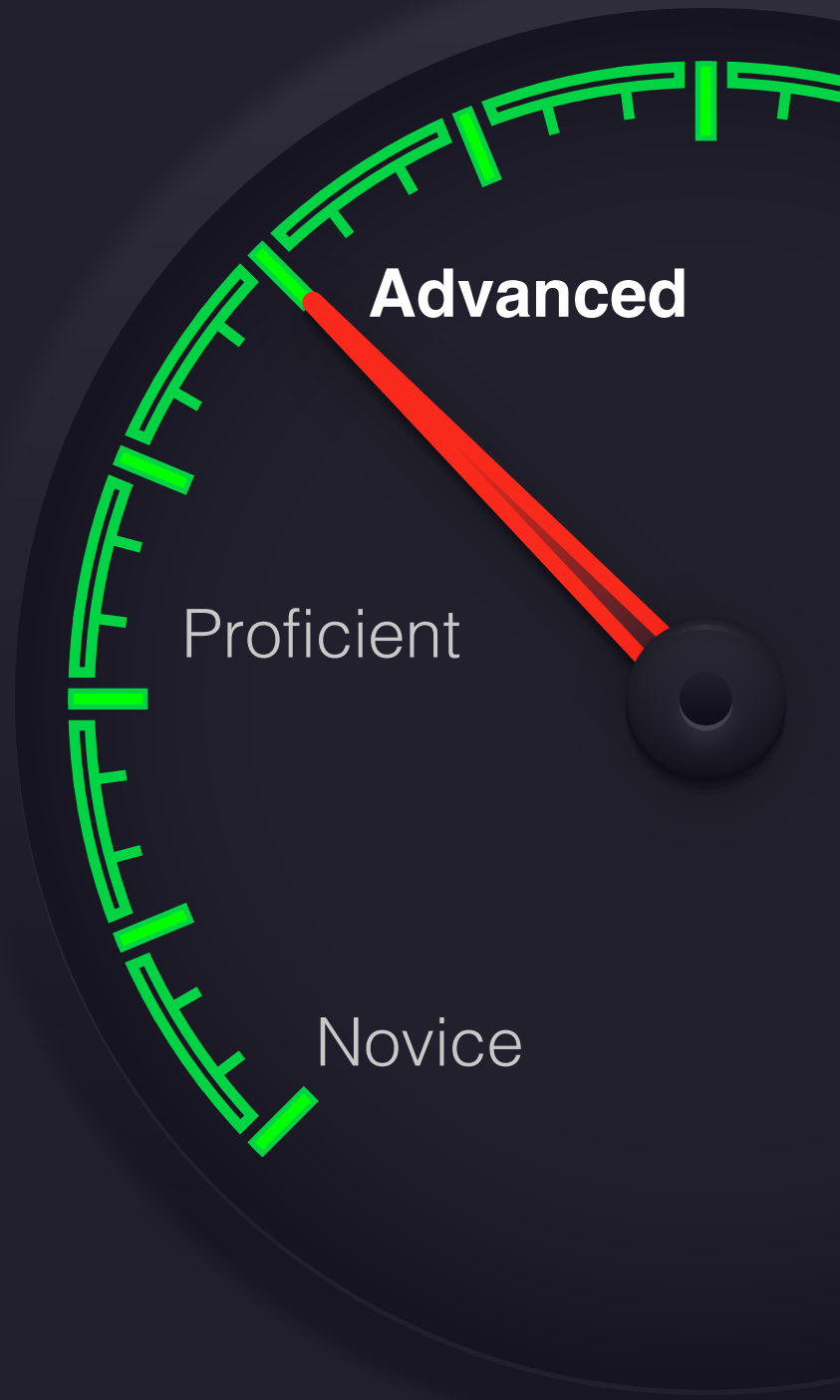
```
instance_id=$(aws ec2 run-instances --image-id ami-12345 \
        --query Reservations[].Instances[].InstanceId \
        --output text)
aws ec2 wait instance-running --instance-ids $instance_id
```

```
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
```

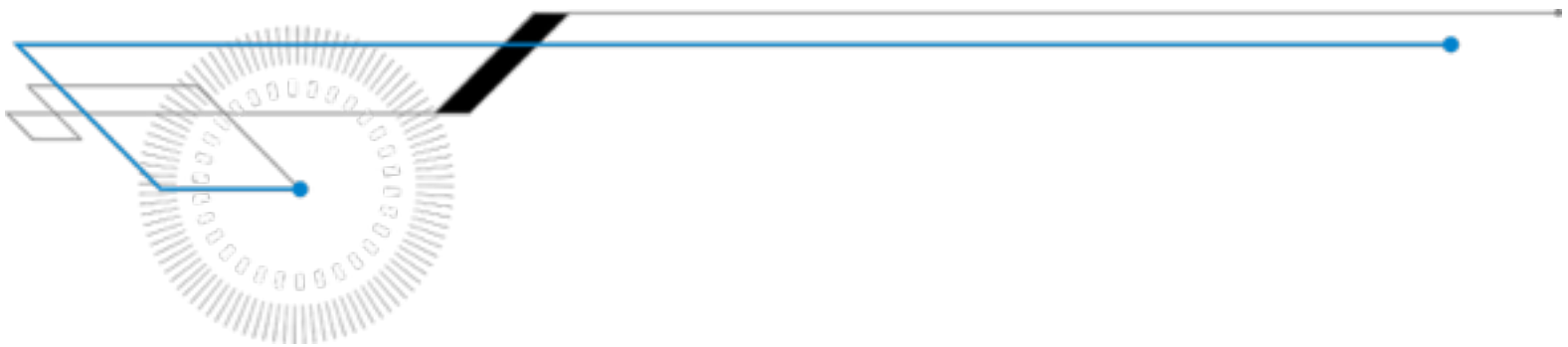


Advanced Scenarios

Looking at advanced AWS CLI features



Templates



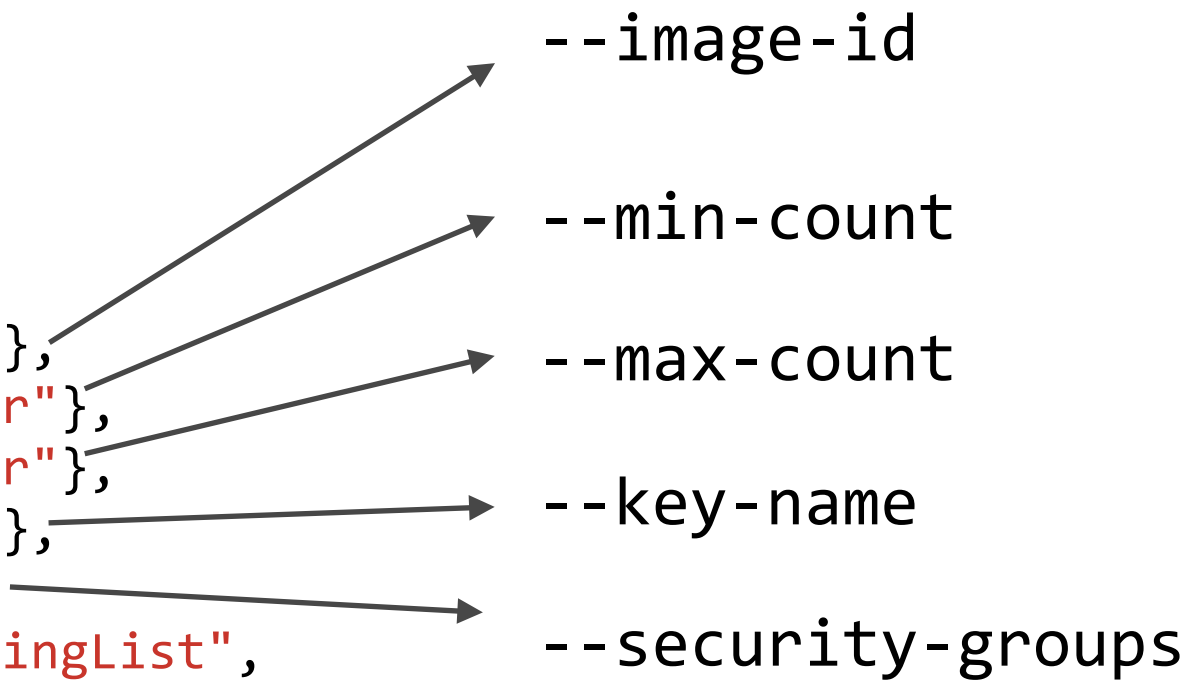
The AWS CLI is data driven

Implementation Details

```
"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"
    },
```

Implementation Details

```
"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"
    },
```



Implementation Details

```
"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
```

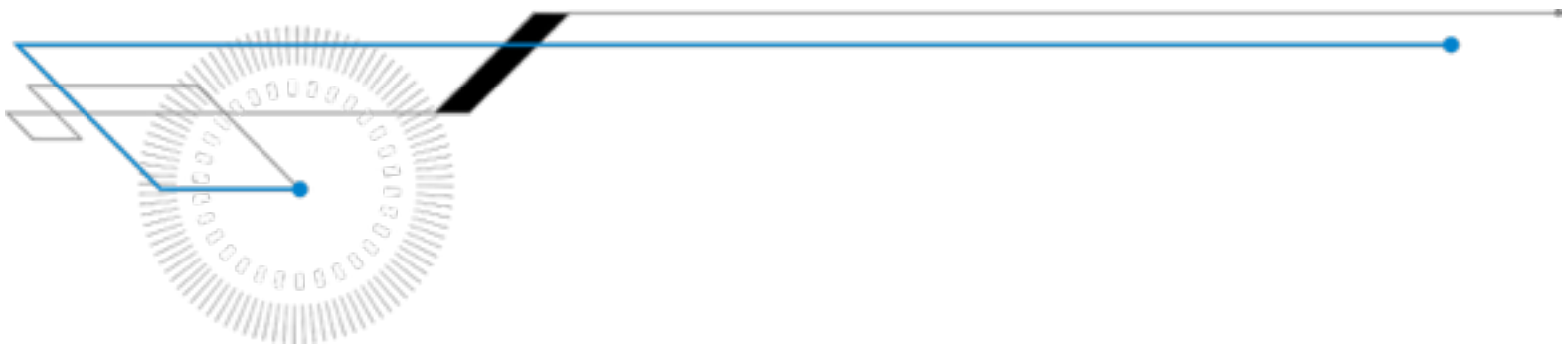
What else can we do?

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

Creating and using JSON templates

DEMO

Credential Providers



CredentialLocator

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

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

CredentialLocator

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

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

CredentialLocator

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

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

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

CredentialLocator

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

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

~/.aws/credentials

~/.aws/config

Amazon EC2
Instance Metadata

CredentialLocator

CredentialLocator

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

~/.aws/credentials

~/.aws/config

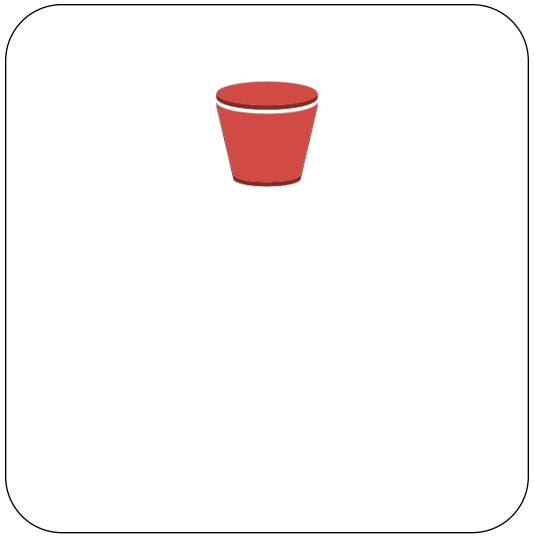
AssumeRole

Amazon EC2
Instance Metadata

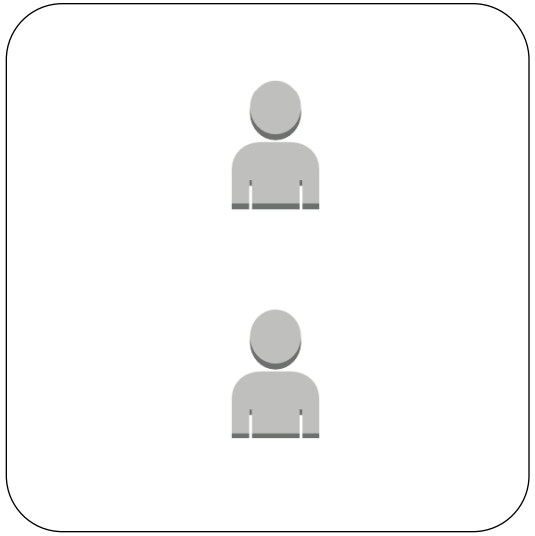
Delegate access to AWS resources using

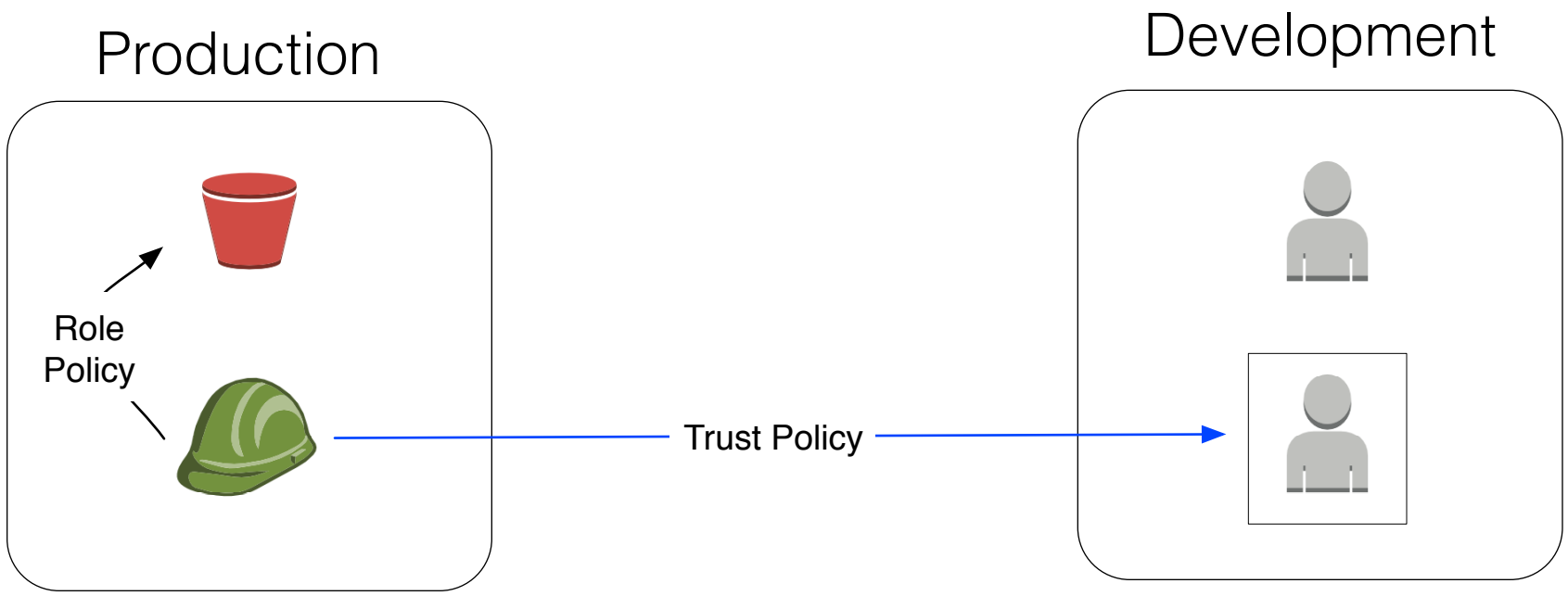
AWS Identity and Access Management (IAM) roles

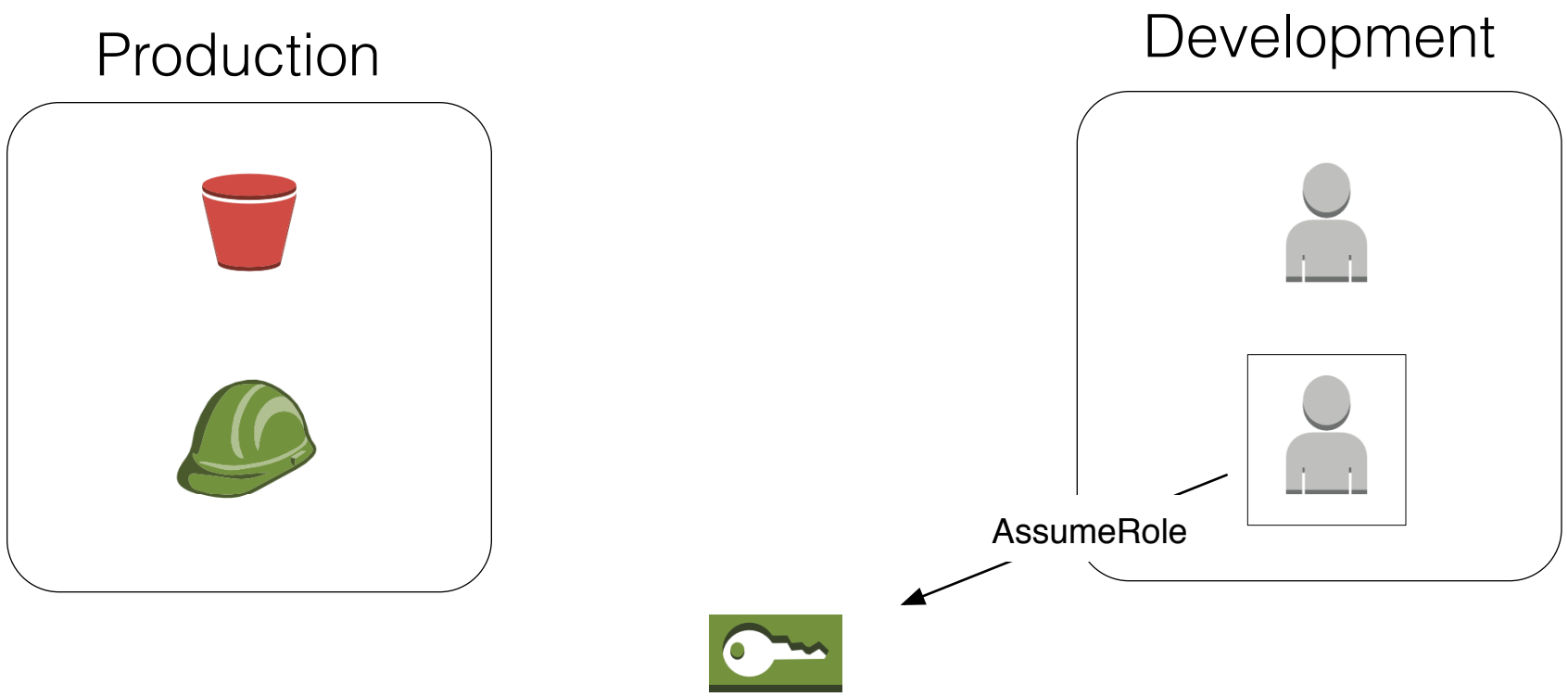
Production

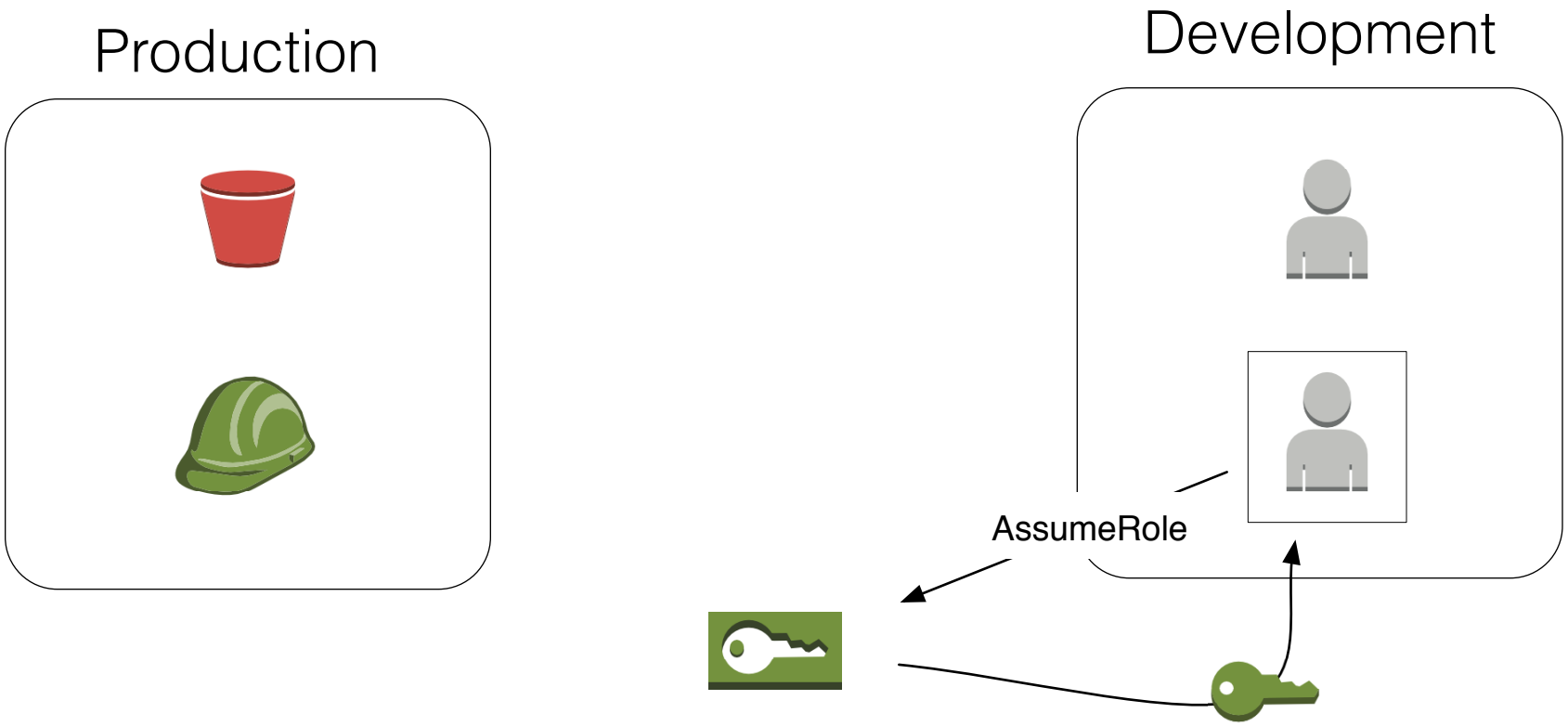


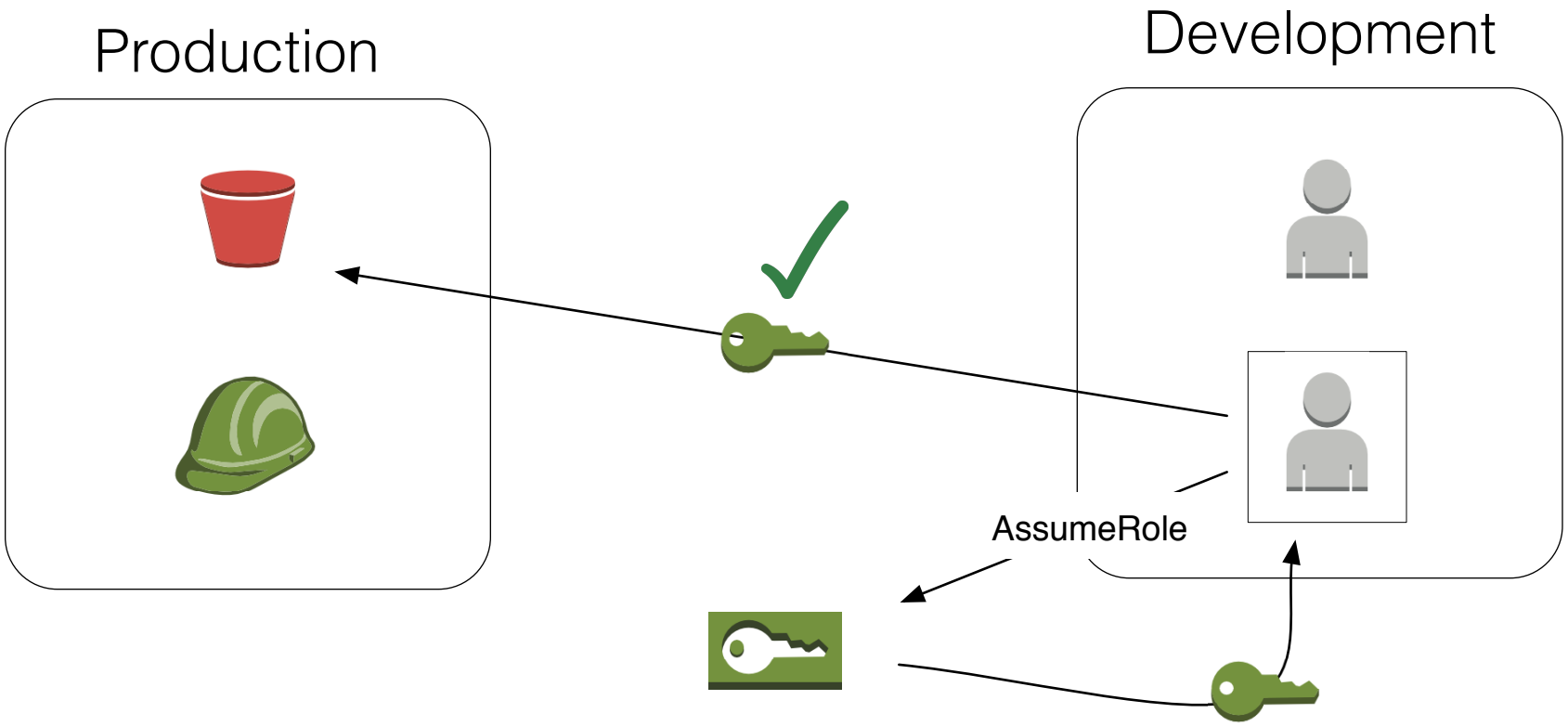
Development




Production
Development
Role
Policy
Trust Policy


Production
Development
AssumeRole


Production
Development
AssumeRole


Production
Development
AssumeRole

```
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

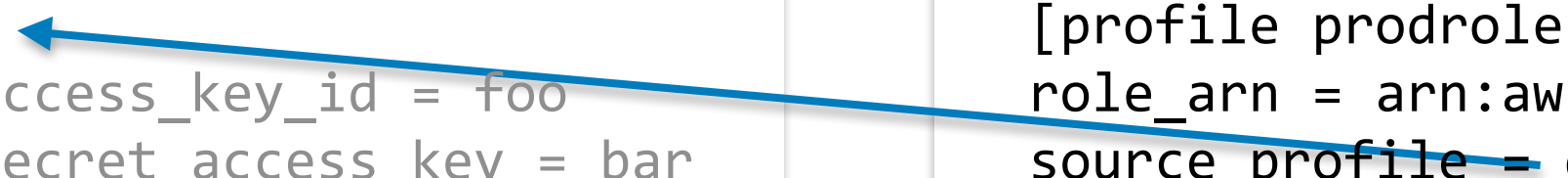
```
[profile prodrole]
role_arn = arn:aws:iam
source_profile = dev
```

~/.aws/credentials

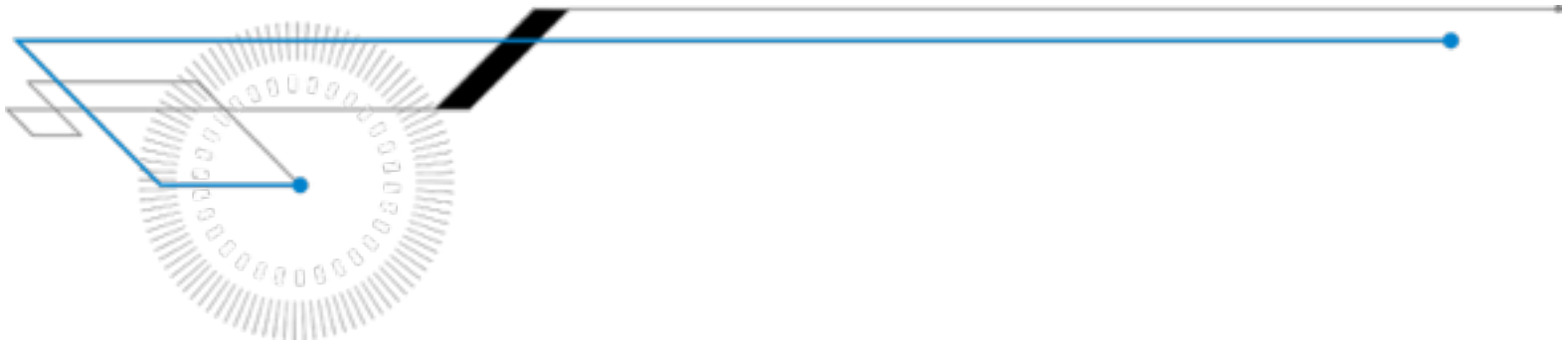
```
[dev]
aws_access_key_id = foo
aws_secret_access_key = bar
```

~/.aws/config

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



Amazon S3 Streaming

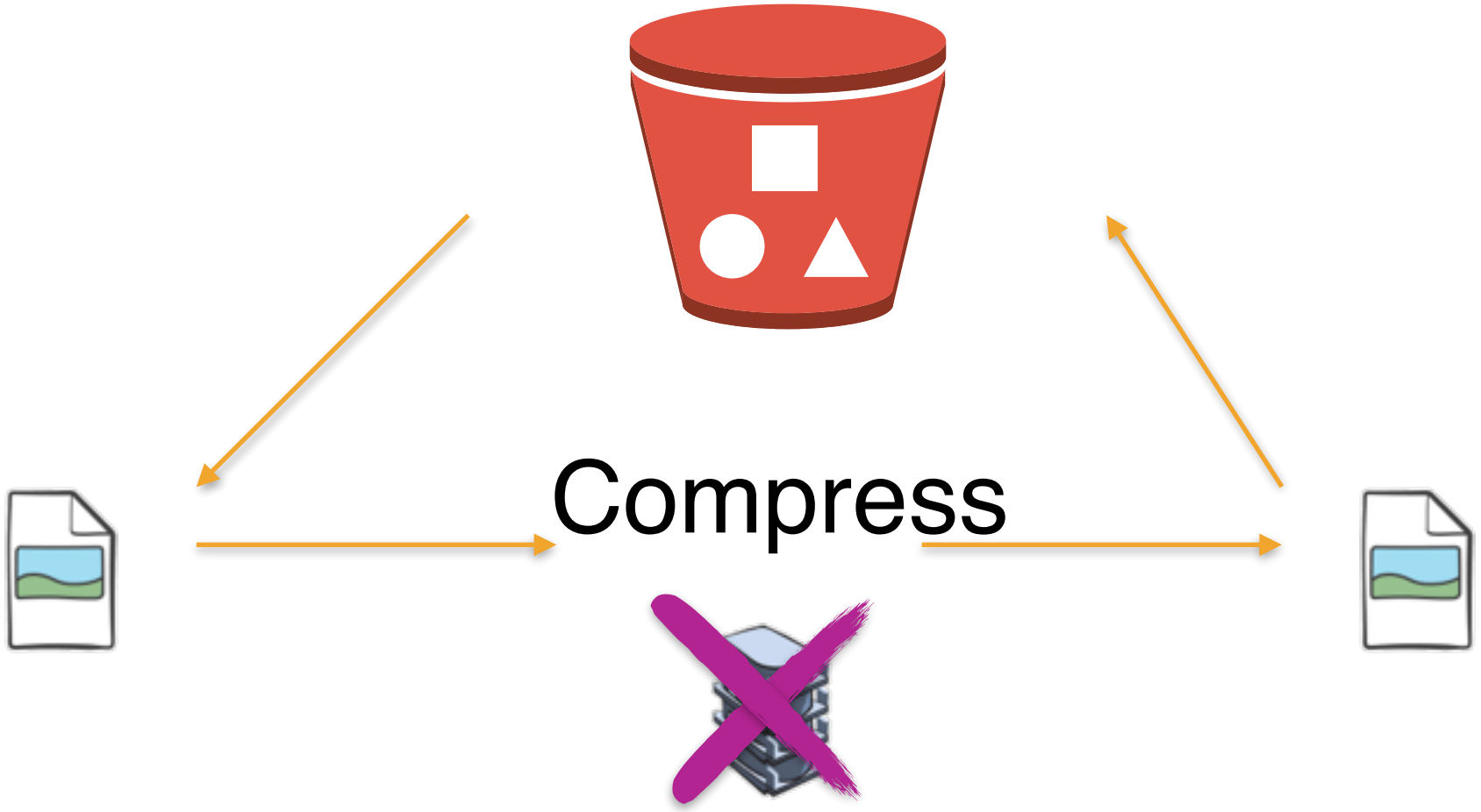


```
aws s3 cp
```

We want to avoid disk

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

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

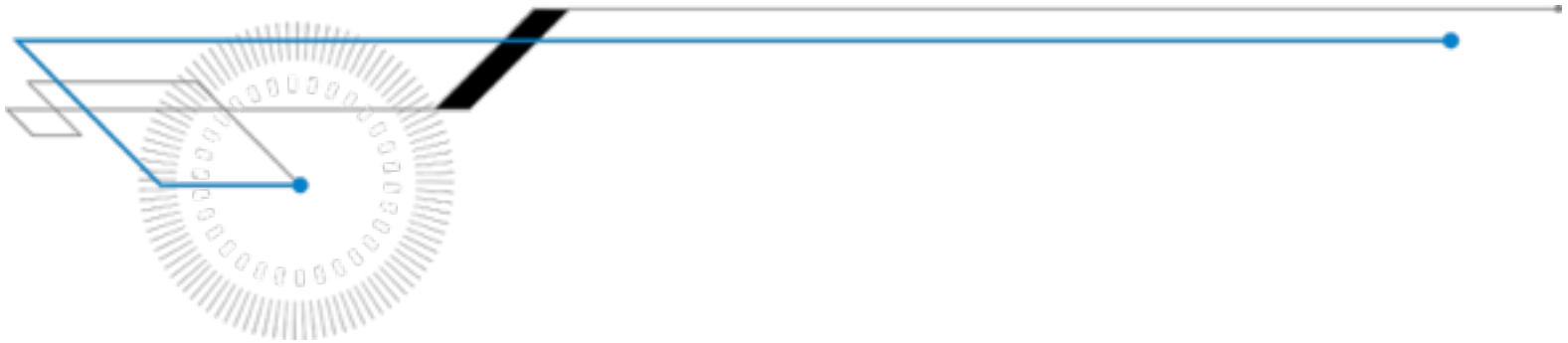

Compress

```
aws s3 cp s3://bucket/key - | gzip | aws s3 cp - s3://bucket/key.gz
```

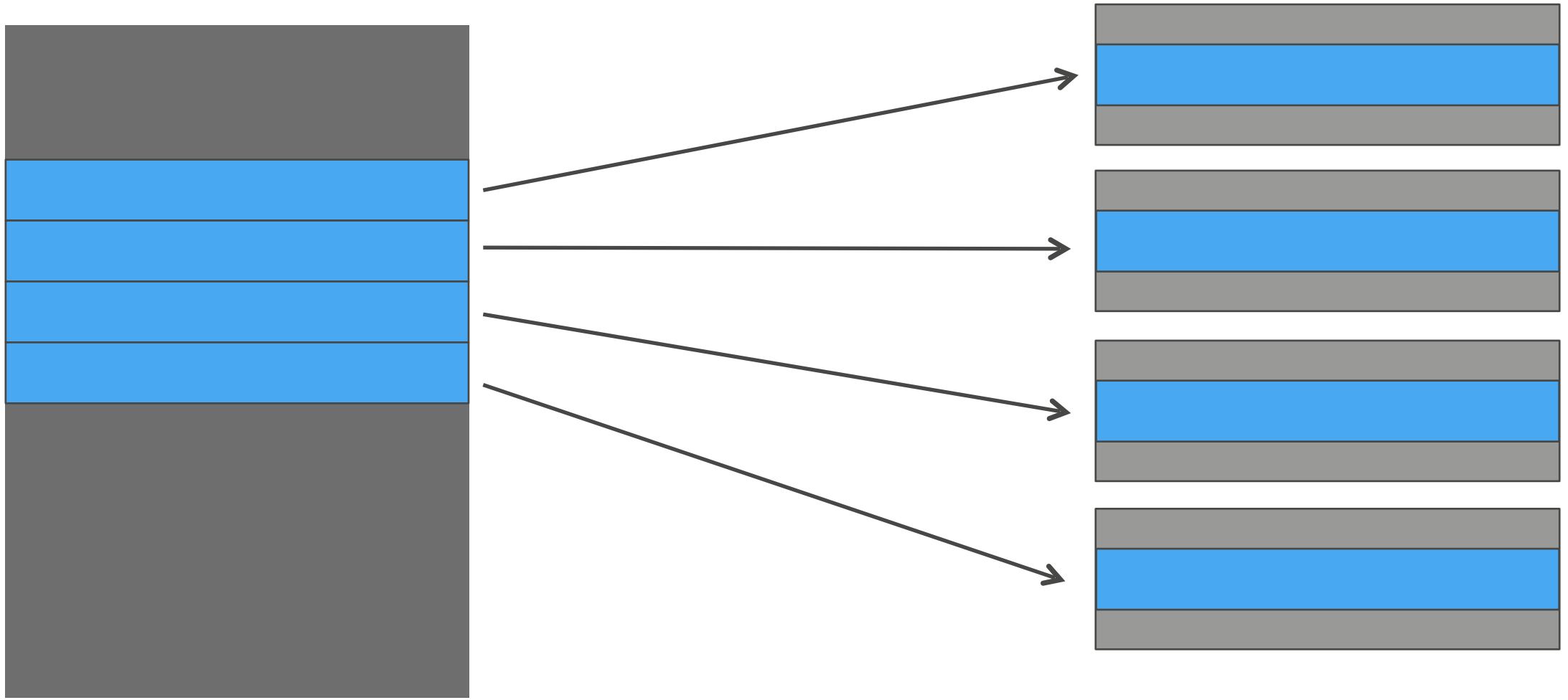
Amazon S3 Streaming

DEMO

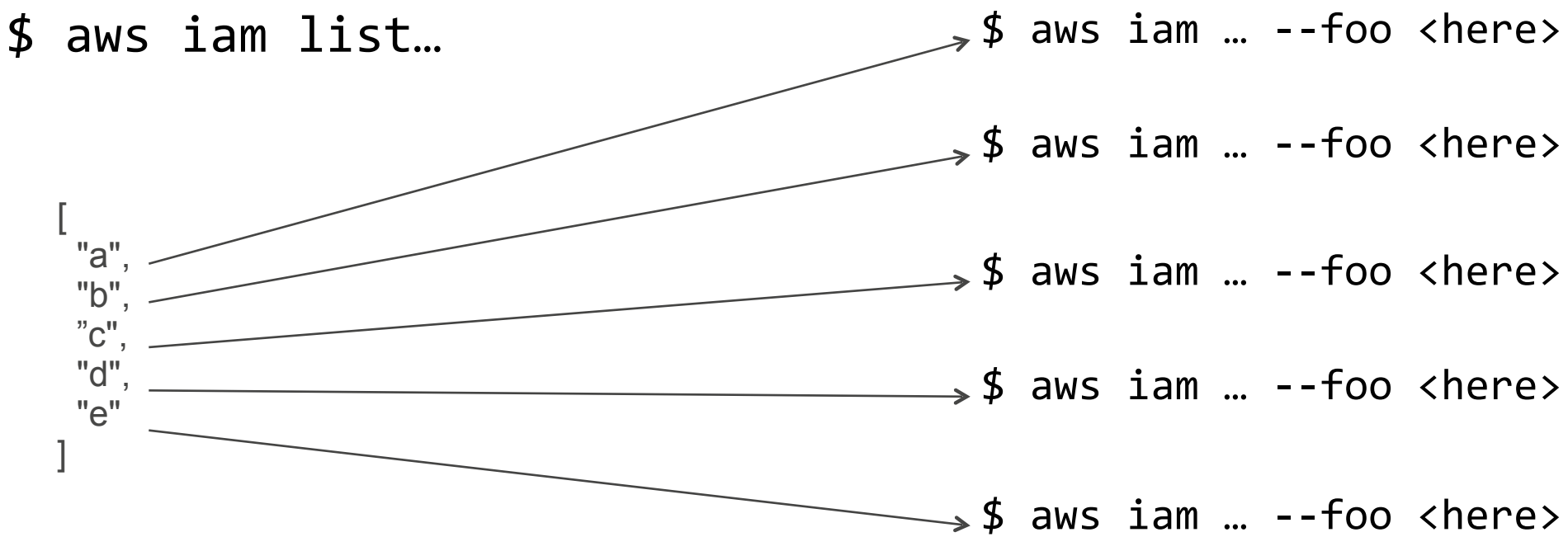
Working in Parallel



Mapping one output to N AWS calls



Mapping one output to N AWS calls

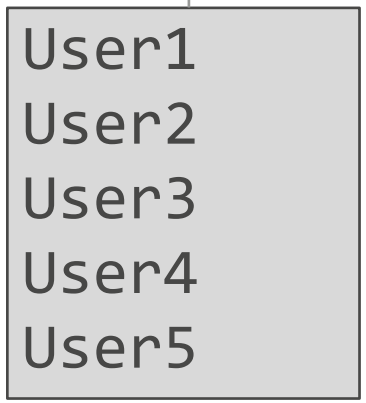


Mapping one output to N AWS calls

```
for name in $(aws iam list-users \
    --query "Users[].[UserName]" --output text); do
  aws iam delete-user --user-name "$name"
done
```

Mapping one output to N AWS calls

```
for name in $(aws iam list-users \
    --query "Users[].[UserName]" --output text); do
  aws iam delete-user --user-name "$name"
done
```



Mapping one output to N AWS calls

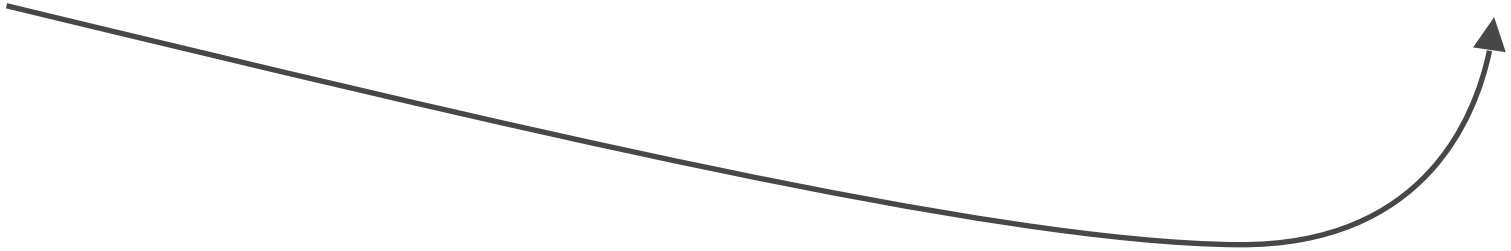
```
for name in $(aws iam list-users \
    --query "Users[].[UserName]" --output text); do
  aws iam delete-user --user-name "$name"
done
```

Mapping one output to N AWS calls

```
aws iam list-users --query "Users[].[UserName]" --output text |
  xargs -I {} -P 10 aws iam delete-user --user-name "{}"
```

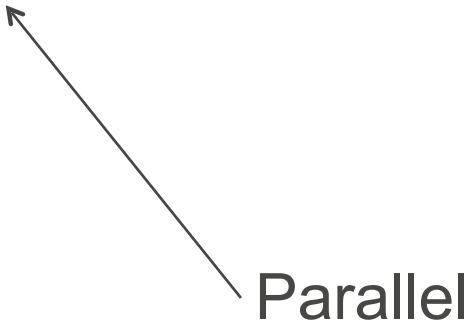
Mapping one output to N AWS calls

```
aws iam list-users --query "Users[].[UserName]" --output text |
  xargs -I {} -P 10 aws iam delete-user --user-name "{}"
```



Mapping one output to N AWS calls

```
aws iam list-users --query "Users[].[UserName]" --output text |
  xargs -I {} -P 10 aws iam delete-user --user-name "{}"
```



Mapping one output to N AWS calls

```
\-+= 72730 james -bash
  \-+- 76343 james xargs -I {} –P 9 aws iam delete-user --user-name {}
    |--- 76348 james aws iam delete-user --user-name user1
    |--- 76349 james aws iam delete-user --user-name user2
    |--- 76350 james aws iam delete-user --user-name user3
    |--- 76351 james aws iam delete-user --user-name user4
    |--- 76352 james aws iam delete-user --user-name user5
    |--- 76353 james aws iam delete-user --user-name user6
    |--- 76354 james aws iam delete-user --user-name user7
    |--- 76355 james aws iam delete-user --user-name user8
    \--- 76356 james aws iam delete-user --user-name user9
```

Mapping one output to N AWS calls

- Use a for loop for functions
- If you're using xargs, use `-I {}`
- Use `xargs -P N` parallel execution
- Use ".[UserName]" instead of ".UserName" to get newline separated output..
- This works because nothing is written to stdout on error

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

Masterclass
Advanced Usage of the AWS CLI

Danilo Poccia

AWS Technical Evangelist

@danilop

danilop