AWS Intermediate

Welcome!

This course is designed to give you some practical experience utilizing various AWS services.

Meet Your Instructor

- Robin Beck

- Lead Instructor for Chef Inc
  - Automation training for CI/CD using cloud on on-prem
  - In-person and online training
  - Course development and certification Exams

Table of Contents

Module 1: DevOps Principles

Develop Intelligence

- DevOps and Infrastructure as code
- Identity Access Management in AWS
- Connecting Resources and IAM, Extra Services and Best Practices
- Working with AWS CLI
- Lab 1: Install AWS CLI, configure, create an instance

Module 2: AWS SDKs

- Advantages of AWS SDKs
- Node SDK Example
- Lab 2: Write a Node script to create an instance and run it

Module 3:

Infrastructure Automation with CloudFormation

- CloudFormation advantage
- CloudFormation structure
- Demo: CloudFormation example and AWS CLI
- Demo: CloudFormation visual web editor
- Lab 3: Create a ELB and auto scaling environment from CloudFormation template/blueprint

Module 4: AWS Databases

- RDS
- DynamoDB
- Creating a database instance
- IAM Role

Module 5: PaaS

» Working with ElasticBeanstalk

Module 6: Serverless

- Serverless with AWS Lambda
- Demo: Building Microservice with Lambda and API Gateway
- Lab 6: Create a lambda CRUD microservice to save data in DB

Why Cloud?

- Elastic, scalable, flexible and operational agile
- Disaster recovery
- Automatic software updates
- Capital-expenditure Free
- Increased collaboration
- Work from anywhere
- Standard and expertise
- Reduced time to market and competitiveness
- Environmentally friendly
- Easy to use
- Benefits of mass economy of scale
- Global delivery faster

Major Cloud Providers:

- Azure
- Google
- AWS

Types of Cloud Computing

- IaaS
- PaaS
- BaaS
- FaaS
- SaaS

AWS Benefits

- One of the first
- Massive scale
- Innovator with news features and services
- Lots of tools - good dev experience
- Almost a standard with lots of expertise, best practices, experts, books, etc.

Best Practices

- Horizontal and vertical scaling
- Redundancy
- Not just EC2- services instead of servers
- Loose coupling
- Stateless
- Automation
- Cost optimization
- Caching

AWS Whitepaper

Develop Intelligence

- Build and deploy faster
- Lower or mitigate risks
- Make informed decisions
- Learn AWS best practices

Whitepaper: pdfs/AWS*Well-Architected*Framework.pdf

Five Pillars of Architecture

Develop Intelligence

1. Security
2. Reliability
3. Performance Efficiency
4. Cost Optimization
5. Operational Excellence

More info

DevOps and Infrastructure as Code

What is DevOps

- Speed of delivery - business value
- Reduce errors (automate)
- Save cost
- Bridge gap between IT Ops and devs - work as one team

Main Principles

- Automate everything
- Version: define infrastructure as code and store in version control system (like app code)
- Ability to deploy and roll back quickly and *often*
- Test app and infrastructure code

Continuous Integration (CI)

Develop Intelligence

For apps:

Dev -> Code version control repository -> Build and Test on CI server -> Deploy to QA -> deploy to prod

For infra:

IT Ops -> Repo -> CI Server: Build images and validate templates (CloudFormation), test APIs -> Deploy

Continuous Delivery (CD)

CI/CD pipeline is automation of CIs. Could include stress testing and performance testing.

Not the same as Continuous Deployment (delivery has manual prod deploy).

Infrastructure as Code

- Repeatability: Humans make mistakes, machines less so (almost 0 when hardware is robust)
- Agility: Deploy quickly and often and roll back quickly and predictably ifneeded
- Auditing: Permissions and ACL with a history

Cloud Automation

- AWS CLI
- SDKs
- CloudFormation
- Others: Ansible, Terraform

Develop Intelligence

What about AWS and its tools?

AWS DevOps and Automation

Develop Intelligence

- Provision environment/infrastructure: AWS CLI, CloudFormation, OpsWorks, Beanstalk
- Configuring servers with AWS: User Data, Docker, Beanstalk, CodeDeploy
- Configuring servers with other tools: Chef, Puppet, SaltStack, Ansible

What we need to do for CI

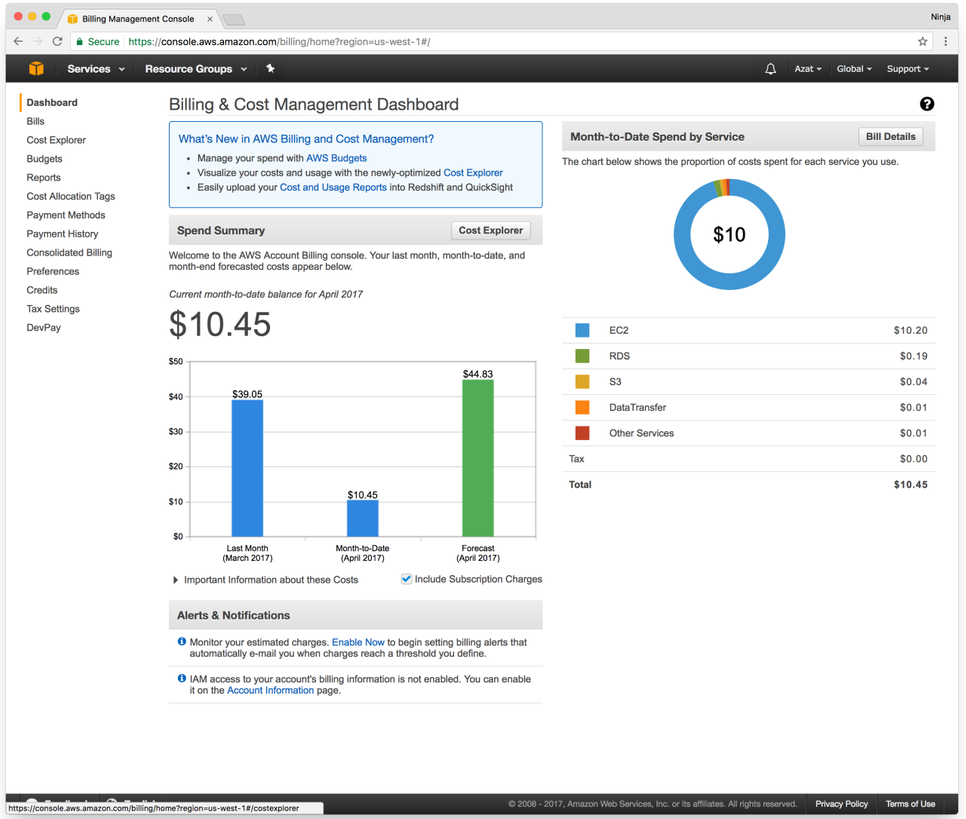
Develop Intelligence

1. Provision environment
2. Deploy code
3. Build
4. Test
5. Verify

Billing and calculator

- SIMPLE MONTHLY CALCULATOR
- Amazon EC2 Pricing
- Amazon S3 Pricing

Billing Management console



What you need or lab 0

Slides & code

Repo: https://github.com/stellarsquall/aws-intermediate

Git clone:

```
git clone https://github.com/stellarsquall/aws-intermediate.git
```

Download with CURL and unzip (create a new folder):

```
curl https://codeload.github.com/stellarsquall/aws-intermediate/zip/master | tar -xv
```

Develop Intelligence

Who has AWS account access? 

AWS account: Free tier

- Sign up for free tier with an email you have access to
- Random verification by email or phone
- Debit/Credit Card for verification and paid services

AWS account: Free tier (cont)

Free tier: https://aws.amazon.com/free, examples:

- EC2: 750 hours of t2.micro (~1 month of 1 EC2) - more than enough for this class and then some more
- S3: 5Gb
- RDS: 750 hours
- Lambda: 1,000,000 requests/mo
- More products!

We have a few pre-requisites—tools you need to install before we can proceed. But where to install?

Install Pre-Reqs Here:

- Host - your dev machine (recommended for Mac and Linux)
- Virtual machine - if you develop in VM (recommended for Windows)
- Remote machine - if you develop in the cloud or if you are setting up CD/CI environment

I develop natively on my dev machine, but you can use another EC2 instance

Pre-Reqs

- AWS Account (requires email + credit/debit card)
- Python 2.7 or 3.x (latest is better)
- AWS CLI: Install with pip or brew or just use a bundle (see all options)
- Node and npm for HTTP server, tools and SKD code (installers)

Good to have tools

- Git mostly for code deploys and Elastic Beanstalk
- Code editor Atom or VS code
- CURL and PuTTY (for Windows)
- Docker deamon/engine - advanced if we have time (instructions)

AWS CLI Check


```
aws --version
```

v1.x - ok

Node and npm Check


```
node --version
```

```
npm --version
```

<= 6.x - ok and

<= 3.x - ok

Lab 0: Installation

- Slides, labs and code https://github.com/azat-co/aws-intermediate
- AWS account
- AWS CLI (pip, brew or bundle)
- Node and npm
- Docker engine

Detailed instructions and link are in labs/0-installs.md

❓ Questions? ❓

AWS CLI

AWS CLI is a very basic way to automate infrastructure and save it in code.

AWS CLI Benefits

- Infrastructure as code: can save in a file and version
- Repeatability: bash script can be run multiple times
- Error free: no need to remember all the steps and configuration for web console
- Fast: no need to click around in the web console
- Can be run from any machine: Will work for CI/CD

Note

\ in a CLI command means a new line - optional and purely for formatting and larger font. \ works the sames in bash/zsh.

Bad font:

```
aws ec2 describe-images --owners amazon --filters "Name=virtualization-type,Values=hvm" "Name=root-device-type,Values=ebs" "Name=name,Values=amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2"
```

Good font:

```
aws ec2 describe-images --owners amazon \
  --filters "Name=virtualization-type,Values=hvm" "Name=root-device-type,Values=ebs" \
  "Name=name,Values=amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2"
```

AWS CLI Usage Pattern

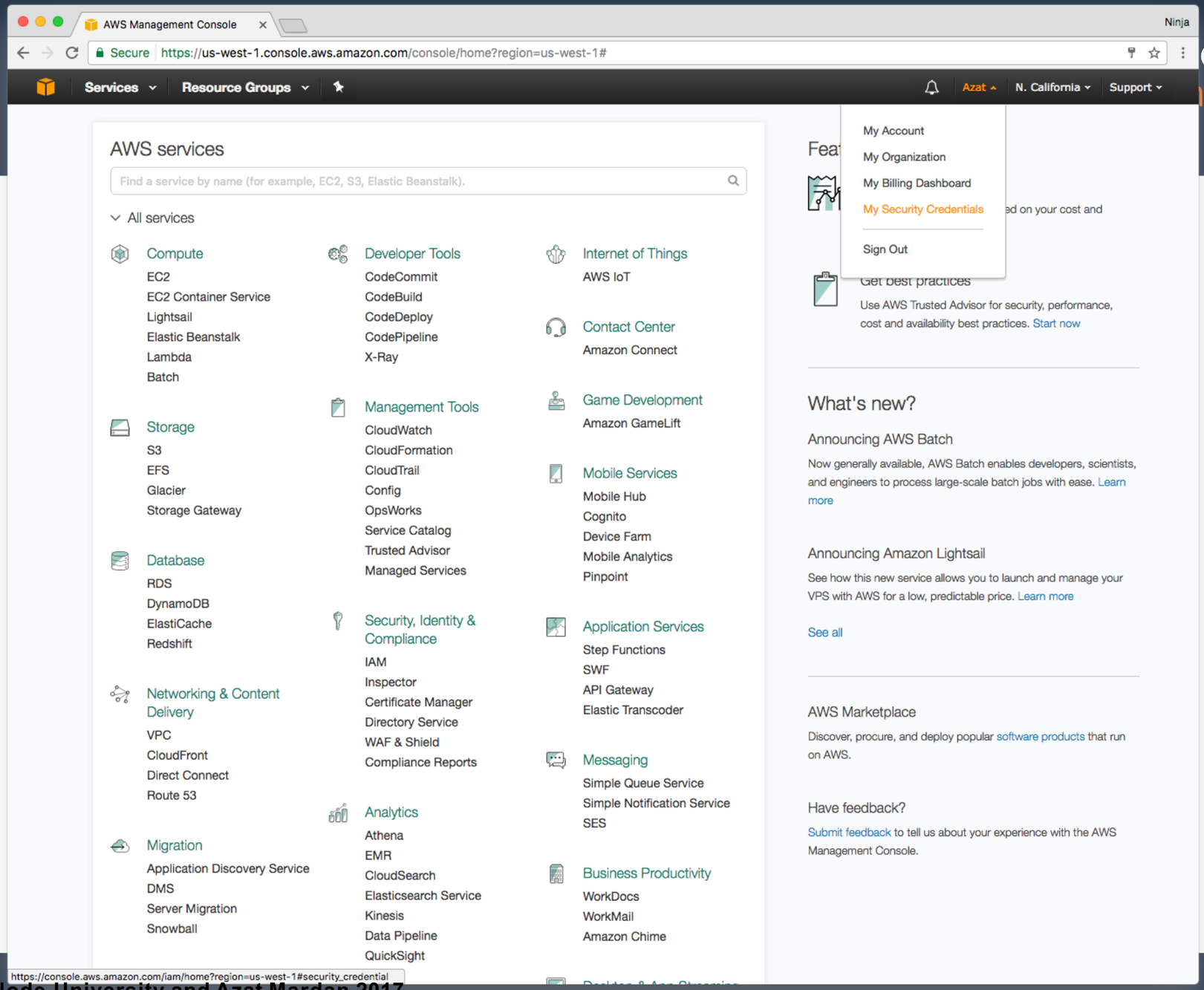

```
aws <command> <subcommand> [options and parameters]
```

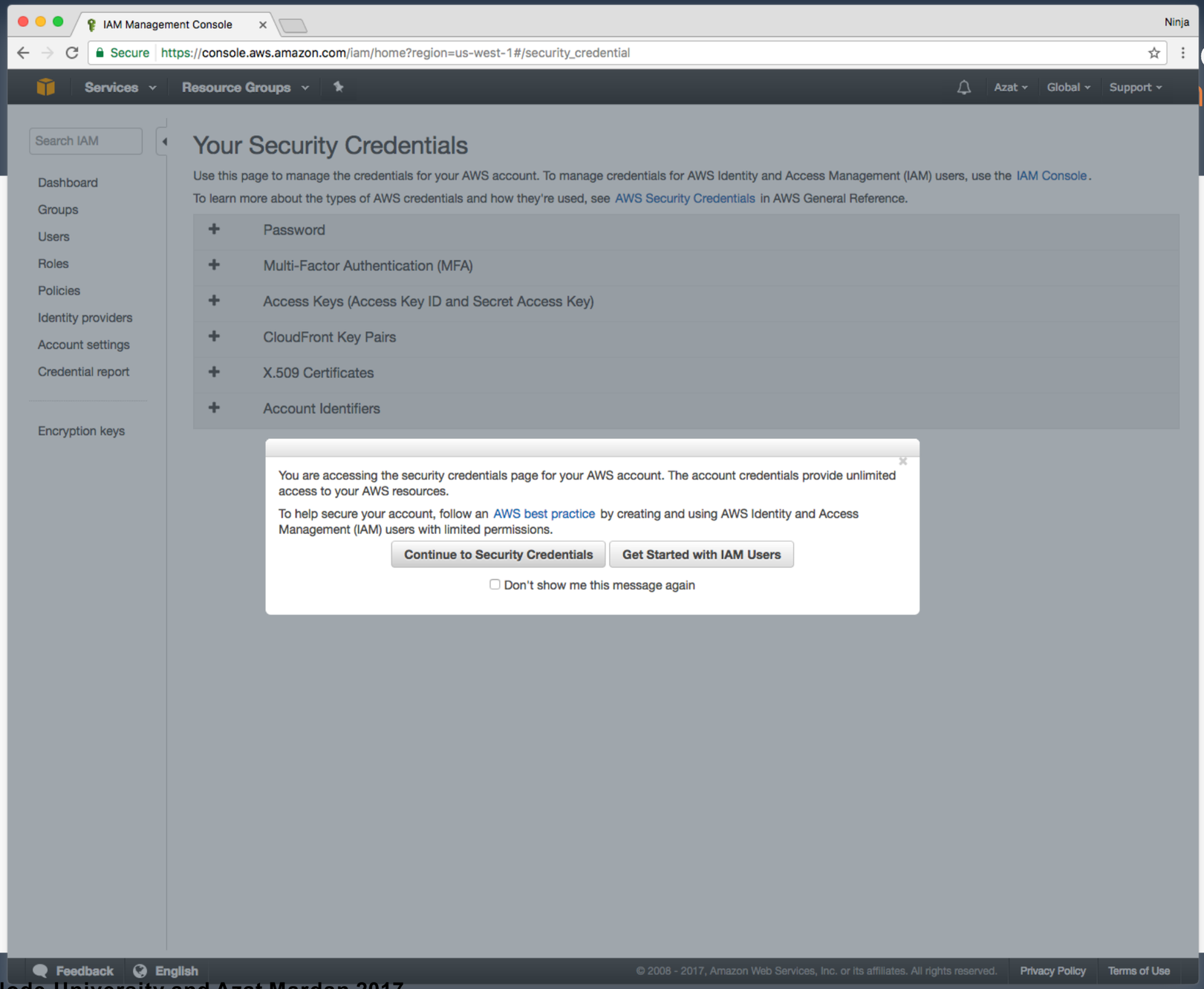
Auth with AWS

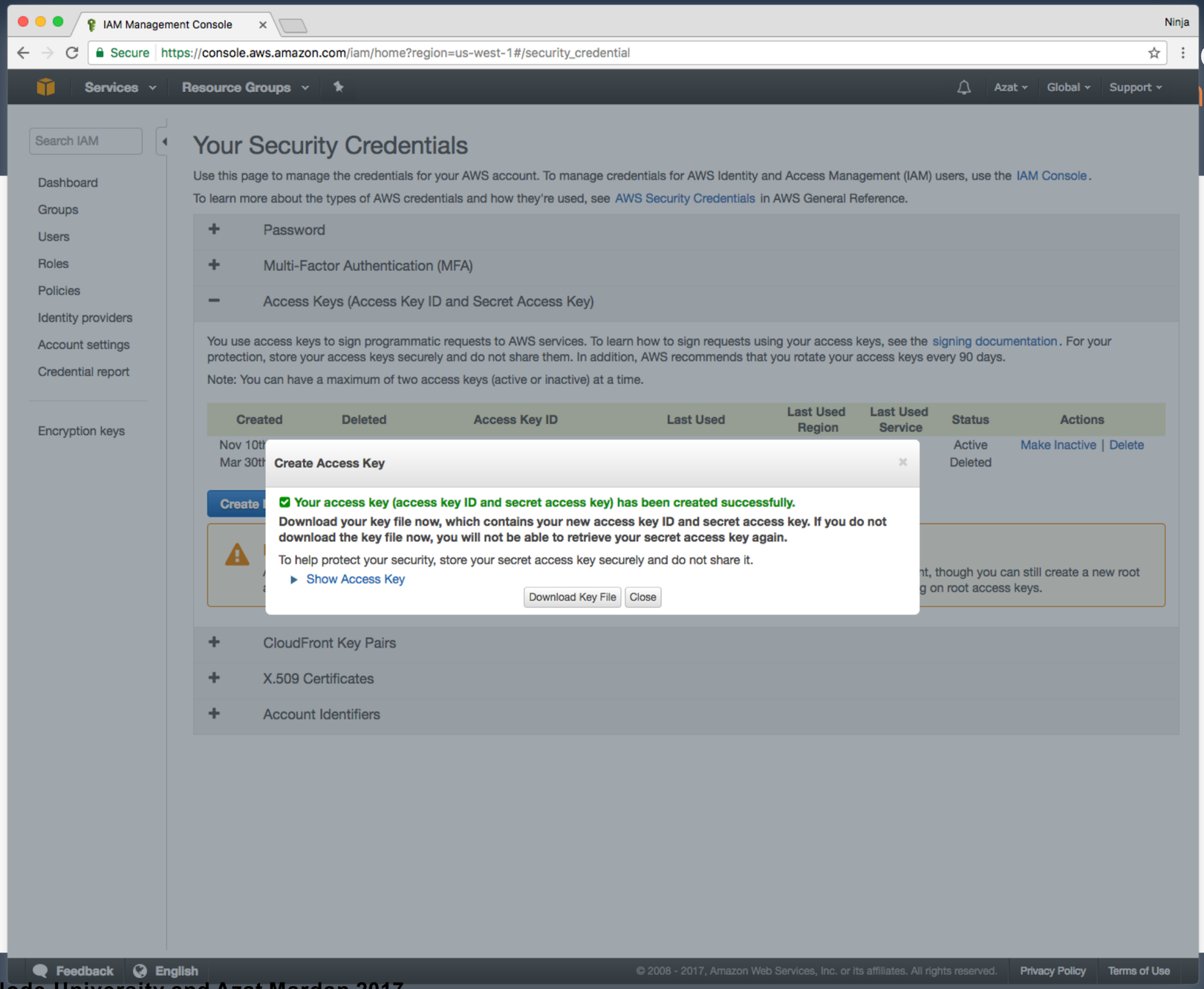
- Access Key ID
- Secret Access Key

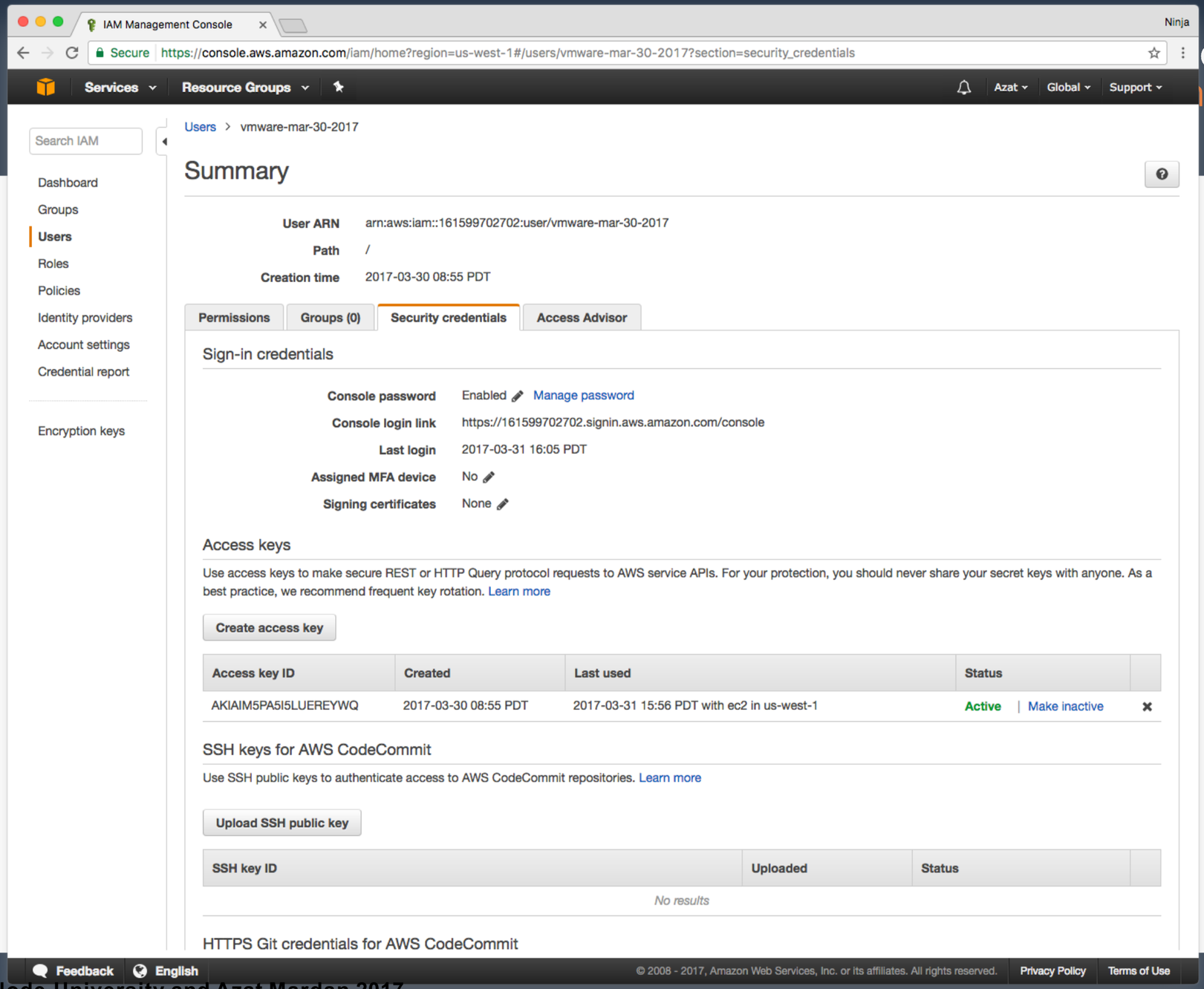
Copy your key and secret (root) or create a new user, give appropriate permissions and copy key and secret for that user (best practice).

Note: You can use AWS CLI to create a user too.









Configure Your CLI

aws configure

1. Provide access key ID
2. Provide secret accesskey
3. Set region to `us-west-1` and output to None or json

Example

```
aws configure
AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-west-1
Default output format [None]: json
```

Getting Help


```
aws help
aws ec2 help
aws ec2 describe-regions help
```

Develop Intelligence

Identity Access Management in AWS

Create User with CLI—Easy!

Develop Intelligence

Create user:

```
aws iam create-user --user-name MyUser
```

Attache policy from a file:

```
aws iam put-user-policy --user-name MyUser --policy-name MyPowerUserRole --policy-document file://C:\Temp\MyPolicyFile.json
```

Or a link:

```
aws iam put-user-policy --user-name MyUser --policy-name MyPowerUserRole --policy-document https://s3.amazonaws.com/checkr3/CC_IAM_FullPolicy.json
```

IAM Policy Example for EC2 Allow All:


```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Stmt1491182980154",
      "Action": "ec2:*",
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

Another IAM JSON file example

List policies for the user to verify:

aws iam list-user-policies --user-name MyUser

Create password to login to web console:

aws iam create-login-profile --user-name MyUser --password Welc0m3!

Create access key:

aws iam create-access-key --user-name MyUser

Create access key response example:

Develop Intelligence


```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

Documentation on IAM

- AWS Identity and Access Management
- AWS Policy Generator

Develop Intelligence

Connecting Resources and IAM

Best IAM Practices

- Lock away your AWS account (root) access keys
- Create individual IAM users
- Use AWS-defined policies to assign permissions whenever possible
- Use groups to assign permissions to IAM users
- Grant least privilege

Best IAM Practices(Cont)

- Configure a strong password policy for your users
- Enable MFA for privileged users
- Use roles for applications that run on Amazon EC2 instances
- Delegate by using roles instead of by sharing credentials

Best IAM Practices (Cont)

- Rotate credentials regularly
- Remove unnecessary credentials
- Use policy conditions for extra security
- Monitor activity in your AWS account

Working with AWS CLI

Getting Started with AWS CLI


```
aws ec2 describe-instances help
aws ec2 run-instances help
awc ec2 create-images help
aws ec2 describe-images help
```

Launch Instance

1. Get image ID
2. Run command

(Amazon Linux AMI IDs are differ from region to region)

Getting Amazon Linux Image ID

Develop Intelligence

- Web console
- `describe-images`
- AWS List

Example of Amazon Linux AMI 2016.09.1 (HVM), SSD Volume Type

Develop Intelligence

```
aws ec2 describe-images --owners amazon \
  --filters "Name=virtualization-type,Values=hvm" "Name=root-device-type,Values=ebs" \
  "Name=name,Values=amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2"
```

Result is "ami-165a0876"

Docs: http://docs.aws.amazon.com/cli/latest/reference/ec2/describe-images.html

Example Output:


```
{
    "Images": [
        {
            "VirtualizationType": "hvm",
            "Name": "amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2",
            "Hypervisor": "xen",
            "ImageOwnerAlias": "amazon",
            "EnaSupport": true,
            "SriovNetSupport": "simple",
            "ImageId": "ami-165a0876",
            "State": "available",
            "BlockDeviceMappings": [
                {
                    "DeviceName": "/dev/xvda",
                    "Ebs": {
                      ...
                    }
                }
            ],
            "Architecture": "x86_64",
            "ImageLocation": "amazon/amzn-ami-hvm-2016.09.1.20170119-x86_64-gp2",
            "RootDeviceType": "ebs",
            "OwnerId": "137112412989",
            "RootDeviceName": "/dev/xvda",
            "CreationDate": "2017-01-20T23:39:56.000Z",
            "Public": true,
            "ImageType": "machine",
            "Description": "Amazon Linux AMI 2016.09.1.20170119 x86_64 HVM GP2"
        }
    ]
}
```

Run instances (really launch or create instances)

```
aws ec2 run-instances --image-id ami-xxxxxxxx \
   --count 1 --instance-type t2.micro \
   --key-name MyKeyPair --security-groups my-sg
```

Note: Need to have security group first (if you don't have it).

Run instances with a subnet:

Develop Intelligence

```
aws ec2 run-instances --image-id ami-{xxxxxxxx} \
   --count 1 --instance-type t2.micro \
   --key-name {MyKeyPair} \
   --security-group-ids sg-{xxxxxxxx} --subnet-id subnet-{xxxxxxxx}
```

Note: Need to have security group and subnet first (if you don't have them).

Develop Intelligence

Working with Security Groups

Create a security group:

```
aws ec2 create-security-group \
    --group-name MySecurityGroup \
    --description "My security group"
```

Add RDP port 3389:

```
aws ec2 authorize-security-group-ingress \
   --group-name my-sg --protocol tcp \
   --port 3389 --cidr 203.0.113.0/24
```

Add SSH port 22:

```
aws ec2 authorize-security-group-ingress \
   --group-name my-sg --protocol tcp \
   --port 22 --cidr 203.0.113.0/24
```

Verify security group:

```
aws ec2 describe-security-groups --group-names my-sg
```

Security Group: Open

Open Everything Example

```
aws ec2 create-security-group --group-name \
   open-sg --description "Open security group"
aws ec2 authorize-security-group-ingress \
   --group-name open-sg --protocol all --port 0-65535 --cidr 0.0.0.0/0
aws ec2 describe-security-groups --group-names open-sg
```

Adding Tags

```
aws ec2 create-tags --resources i-{xxxxxxxx} \
   --tags Key={Name},Value={MyInstance}
```

Replace {xxx}, {Name} and {MyInstance}

View instances

```
aws ec2 describe-instances
```

Stopping, starting, and terminating

Develop Intelligence

```
aws ec2 stop-instances --instance-ids i-{xxxxxxxx}
aws ec2 start-instances--instance-ids i-{xxxxxxxx}
aws ec2 terminate-instances --instance-ids i-{xxxxxxxx}
```

Note: after stop you can start, after terminate no.

Working with Key Pairs


```
aws ec2 create-key-pair --key-name {MyKeyPair} \
   --query 'KeyMaterial' --output text > {MyKeyPair}.pem
aws ec2 describe-key-pairs --key-name {MyKeyPair}
aws ec2 delete-key-pair --key-name {MyKeyPair}
```

{MyKeyPair} is a string name, e.g., my-aws-dev.

Auto Startup

- `init.d` or CloudInit for Ubuntu+Debian and other like CentOS with additional installation
- User Data
- Command

Note: More on User Data is in the AWS Intro course

Startup Script

```
#!/bin/bash
curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.32.0/install.sh | bash
. ~/.nvm/nvm.sh
nvm install 6
node -e "console.log('Running Node.js ' +process.version)"
echo "require('http').createServer((req,res) => {
  res.end('hello world')
}).listen(3000, (error)=>{
  console.log('server is running on 3000')
})
" >> index.js
node index.js
```

Shell Script and UserData Example

LAMP Stack (Apache httpd, MySQL and PHP) for Amazon Linux:

```
#!/bin/bash
yum update -
y
yuminstall -y httpd24 php56 mysql55-server php56-
mysqlnd service httpd start
chkconfig httpd
on groupadd www
usermod -a -G www ec2-
user chown -R root:www
/var/www chmod 2775
/var/www
find /var/www -type d -exec chmod 2775 {} +
find /var/www -type f -exec chmod 0664
{} +
echo "<?php phpinfo(); ?>" > /var/www/html/phpinfo.php
echo "<?php echo 'Hello World!' ?>" > /var/www/html/index.php
```

User Data in run-instances

Develop Intelligence

You can supply base64 encoded string, normal string or a file. ami-9e247efe is Amazon Linux AMI for us-west-1:

```
aws ec2 run-instances  --image-id  ami-9e247efe \
   --count  1 --instance-type  t2.micro \
   --key-name  MyKeyPair \
   --security-groups  MySecurityGroup   \
   --user-data  file://my_script.txt
```

Note: You can only run user-data once on launch (run-instances). Updating user data on existing instance will NOT run the User Data script.

More info on User Data:

http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/user-data.html

Lab 1: Power to AWS CLI

Task: Install AWS CLI, configure, create an instance with apache httpd via AWS CLI and no SSH, make the HTML page (hello world) visible in the browser *publicly*

Detailed instructions and link are in labs/1-hello-world-aws-cli.md

Time to finish: 15 min 

Develop Intelligence

Module 2: AWS SDKs

How to access and work with AWS platform from within your application?

SDKs!

Advantages of SDKs

- Automate anything
- Build your own clients or interfaces for AWS
- No need to create HTTP requests and worry about payloads, formats, headers
- Work in your favorite environment: Java, Python, Node and many more

Supported services

- Amazon S3
- Amazon EC2
- DynamoDB
- Many more!

What languages

- Android
- Browser
- iOS
- Java
- .NET
- Node.js
- PHP
- Python
- Ruby
- Go

Node SDK

```
mkdir aws-node-sdk-test
cd aws-node-sdk-test
npm init -y
npm i -SE aws-sdk
```

Credentials

- Home directory
- Environment variables
- JavaScript/Node or JSON file

Pick just one

Credentials in Home Directory

Develop Intelligence

~/.aws/credentials or C:\Users\USER_NAME\.aws\credentials for Windows users

```
[default]
aws_access_key_id = YOUR_ACCESS_KEY_ID
aws_secret_access_key = YOUR_SECRET_ACCESS_KEY
```

(region goes into Node file in new AWS.EC2 call)

Env variables

AWS_ACCESS_KEY_ID=key

AWS_SECRET_ACCESS_KEY=secre

t AWS_REGION=us-west-1

JSON Config for Creds

```
{
  "accessKeyId": "your-access-key-here",
  "secretAccessKey": "your-secret-key-here",
  "region": "us-west-1"
}
```

EC2 Example

describe.js


```
// Load the SDK for JavaScript
const AWS = require('aws-sdk')
// Load credentials and set region from JSON file
// AWS.config.loadFromPath('./config.json')
// Create EC2 service object
const ec2 = new AWS.EC2({apiVersion: '2016-11-15', region:'us-west-1'})
```

```
const params = {
  // DryRun: true || false, Filters: [
    {
      Name: 'endpoint', Values: [
        'ec2.us-west-1.amazonaws.com',
        /* more items */
      ]
    },
    /* more items */
  ],
  RegionNames: [ 'us-west-1',
    /* more items */
  ]
}
```

```
// Describe region
ec2.describeRegions(params, function(err, data) {
    if (err) return console.log('Could not describe regions', err)  console.log(data)
    const imageParams = {  Owners: ['amazon'], Filters: [{
        Name:'virtualization-type', Values: ['hvm']
      }, {
        Name:'root-device-type', Values: ['ebs']
      }, {
        Name: 'name',
        Values: ['amzn-ami-hvm-2017.03.0.*-x86_64-gp2']
      }]
    }
    ec2.describeImages(imageParams, (err, data)=>{
      if (err) return console.log('Could not describe regions', err)  console.log(data)
    })
})
```

How to run it?

```
cd code/sdk

node describe.js
```

Create and open provision-infra.js:


```
// Load the SDK for JavaScript
const AWS = require('aws-sdk')
// Load credentials and set region from JSON file
AWS.config.loadFromPath('./config.json')
```

```
// Create EC2 service object
var ec2 = new AWS.EC2({apiVersion: '2016-11-15'})
//const ec2 = new AWS.EC2({apiVersion: '2016-11-15', region:'us-west-1'})
const fs=  require('fs')
var params = {
    ImageId: 'ami-7a85a01a', // us-west-1 Amazon Linux AMI 2017.03.0 (HVM), SSD Volume Type
    InstanceType: 't2.micro',
    MinCount: 1,
    MaxCount: 1,
    UserData: fs.readFileSync('./user-data.sh', 'base64')
}
```

```
// Create the instance ec2.runInstances(params, function(err, data) {
    if (err) {
        console.log('Could not create instance', err) return
    }
    var instanceId = data.Instances[0].InstanceId console.log('Createdinstance',     instanceId)
    // Add tags to the instance
    params = {Resources: [instanceId], Tags: [
        {
            Key: 'Role',
            Value: 'aws-course'
        }
    ]}
    ec2.createTags(params, function(err) {
        console.log('Tagging instance', err ? 'failure' : 'success')
    })
})
```

Running Node scripts

`node provision-infra.js`

Example: code/sdk

Run:

```
cd code/sdk
node provision-infra.js
Created instance i-0261a29f670faade4
Tagging instance success
```

Describe:

```
aws ec2 describe-instances --instance-ids i-0261a29f670faade4
AWS_DEFAULT_OUTPUT=table aws ec2 describe-instances --instance-ids i-0261a29f670faade4
aws ec2 describe-instances --instance-ids i-0261a29f670faade4 \
  --query "Reservations[0].Instances[0].PublicDnsName"
```

Feeding Data to Node

Develop Intelligence

```
node provision-infra.js ./user-data-qa.js
```

```
const userDataFile = process.argv[2]
//...
var params = {
    ImageId: 'ami-7a85a01a', // us-west-1 Amazon Linux AMI 2017.03.0 (HVM), SSD Volume Type
    InstanceType: 't2.micro',
    MinCount: 1,
    MaxCount: 1,
    UserData: fs.readFileSync(userDataFile, 'base64')
}
```

Lab 2: Node SDK Runs EC2

Task: Write a Node script to create an instance with Node hello world (use User Data), and run it. Make sure you see the Hello World via public DNS.

Detailed instructions and link are in labs/02-sdk-runs-ec2.md

Time to finish: 10 min

Develop Intelligence

Module 3: Cloud Infrastructure Automation with CloudFormation

Declarative vs. Imperative

Develop Intelligence

TL;DR:

Declarative - what I want, declare the end-state
Imperative - what to do, describe the steps

Declarative requires that users specify the end state of the infrastructure they want, while imperative configures systems in a series of actions.

(AWS CLI and SDK are imperative.)

CLI and SDKs are imperative

Develop Intelligence

- Not simple to understand the end result
- Race conditions
- Unpredictable results

Meet CloudFormation!

What is CloudFormation

Develop Intelligence

- Special format in a JSON or YAML file
- Declarative service
- Visual web editor

Samples

CloudFormation advantages

Develop Intelligence

- Declarative and Flexible
- Easy to Use
- Infrastructure as Code 0 1 0 1
- Supports a Wide Range of AWS Resources
- Customized via Parameters
- Visualize and Edit with Drag-and-Drop Interface
- Integration Ready

CloudFormation Example:

S3 Bucket

```
{
    "Resources" : {
        "HelloBucket" : {
            "Type" : "AWS::S3::Bucket"
        }
    }
}
```

YAML

```
Resources:
    HelloBucket:
        Type:  AWS::S3::Bucket
```

Where to put JSON or YAML?

Develop Intelligence

- CLI
- Web console
- SDK
- REST API calls

AWS CLI create-stack

aws cloudformation create-stack --stack-name myteststack --template-body file:////home//local//test//sampletemplate.json

It will give you stack ID which you can use later to check on the status of creation.

CloudFormation structure

Develop Intelligence

- Version
- Description
- Resources
- Parameters
- Mappings
- Outputs

CloudFormation Resources

Develop Intelligence

- Resource must have a type of this format `AWS::ProductIdentifier::ResourceType`. See all resource types.
- Some resources like S3 have defaults but others like EC2 will require more properties (image ID)
- You can get real property value with Ref function (ID, IP, etc.)

Develop Intelligence

Let's use ref function. See list of ref functions.

EC2 with a security group

```
{
  "Resources" : {  "Ec2Instance"  : {
      "Type" :  "AWS::EC2::Instance",  "Properties"  : {
        "SecurityGroups" : [ { "Ref" :  "InstanceSecurityGroup" } ],   "KeyName"  :  "my-aws-course",
        "ImageId"  :  "ami-9e247efe"
      }
    },
    "InstanceSecurityGroup"  : {
      "Type" :  "AWS::EC2::SecurityGroup",  "Properties"  : {
        "GroupDescription" :  "Enable SSHaccess via port 22",  "SecurityGroupIngress"  : [ {
          "IpProtocol"  :  "tcp",
          "FromPort"  :  "22",
          "ToPort"  :  "22",
          "CidrIp"  :  "0.0.0.0/0"
        } ]
      }
    }
  }
}
```

Develop Intelligence

For other attributes not returned by ref, there's Fn::GetAtt

```
{ "Fn::GetAtt" : [ "logicalNameOfResource", "attributeName" ] }
```

See reference.

my-aws-course keypair must exist first!

```
"Resources" : {
  "Ec2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "SecurityGroups" : [ { "Ref" : "InstanceSecurityGroup" } ],
      "KeyName" : "my-aws-key",
      "ImageId" : "ami-9e247efe"
    }
  },
```

Develop Intelligence

`my-aws-key` must exist before running this template... can we provide it later? Yes, it's a template!

Develop Intelligence

We can use Parameters

Key parameter

Develop Intelligence

(key is provided on stack creation)

```
{
  "Parameters" : {
    "KeyNameParam" : {
      "Description" : "The EC2 Key Pair to allow SSH access to the instance",
      "Type" : "AWS::EC2::KeyPair::KeyName"
    }
  },
```

Resource EC2

Develop Intelligence

(created with the key from KeyNameParam)

```
"Resources" : {
  "Ec2Instance" : {
    "Type" : "AWS::EC2::Instance",
    "Properties" : {
      "SecurityGroups" : [{"Ref"    : "InstanceSecurityGroup" }, "MyExistingSecurityGroup" ],
      "KeyName" : { "Ref" : "KeyNameParam"},
      "ImageId" : "ami-7a11e213"
    }
  },
```

Resource SecurityGroup (created by CloudFormation)

```
    "InstanceSecurityGroup" : {
      "Type" : "AWS::EC2::SecurityGroup",
      "Properties" : {
        "GroupDescription" : "Enable SSH access via port 22",
        "SecurityGroupIngress" : [ {
          "IpProtocol" : "tcp",
          "FromPort" : "22",
          "ToPort" : "22",
          "CidrIp" : "0.0.0.0/0"
        } ]
      }
    }
  }
}
```

Providing parameters in CLI

Develop Intelligence

```
aws cloudformation create-stack --stack-name myteststack \
  --template-body file:////home//local//test//sampletemplate.json \
  --parameters ParameterKey=KeyNameParam,ParameterValue=my-aws-key \
  ParameterKey=SubnetIDs,ParameterValue=SubnetID1\\,SubnetID2
```

WordPress CloudFormation Parameters Example


```
"Parameters": {   "KeyNameParam": {
    "Description" :  "Name of an existing EC2 KeyPair to enable SSH access into the WordPress web server",   "Type":
    "AWS::EC2::KeyPair::KeyName"
  },
  "WordPressUser": {   "Default":  "admin",
    "NoEcho": "true",
    "Description" :  "The WordPress database admin account user name",  "Type":  "String",
    "MinLength":  "1",
    "MaxLength":  "16",
    "AllowedPattern"  :  "[a-zA-Z][a-zA-Z0-9]*"
  },
  "WebServerPort": {   "Default":  "8888",
    "Description" :  "TCP/IP port for the WordPress web server",   "Type":  "Number",
    "MinValue":  "1",
    "MaxValue":  "65535"
  }
}
```

Pseudo Parameters

Resolved by CloudFormation, e.g., AWS::Region

Other functions

- Fn::FindInMap
- Fn::Base64
- Conditional: Fn::And, Fn::Equals, Fn::If, Fn::Not, Fn::Or

See the full list in Intrinsic Function Reference.

Mappings

What and why Mappings

Mappings is for specifying conditional values

Simple Parameters -> Mappings -> Complex Values

Example: Getting AMI ID (differs from region to region for the same image)

Mappings example: AMI IDs based on regions:

Develop Intelligence

```
"Mappings" : {
  "RegionMap" : {  "us-east-1"  : {
        "AMI" : "ami-76f0061f"
    },
    "us-west-1"  : {
        "AMI" : "ami-655a0a20"
    },
    "eu-west-1"  : {
        "AMI" : "ami-7fd4e10b"
    },
    "ap-southeast-1"  : {
        "AMI" : "ami-72621c20"
    },
    "ap-northeast-1"  : {
        "AMI" : "ami-8e08a38f"
    }
  }
},
```

Find AMI ID based on region using mappings:


```
"Resources" : {
    "Ec2Instance" : {
      "Type" : "AWS::EC2::Instance",
      "Properties" : {
        "KeyName" : { "Ref" : "KeyNameParam" },
        "ImageId" :{"Fn::FindInMap" : [ "RegionMap",   { "Ref" : "AWS::Region" }, "AMI" ]},
        "UserData" : { "Fn::Base64" : "80" }
      }
    }
  }
}
```

Lab 3: Form the Cloud

Develop Intelligence

Task: Create an ELB, security group and auto scaling environment from CloudFormation template/blueprint; load/stress test it to see auto increase

You can use blueprint from code/cloudformation/AutoScalingMultiAZWithNotifications.json or one from AWS

Detailed instructions and link are in labs/03-form-the-cloud.md

Time to finish: 20min

Develop Intelligence

Module 4: AWS Databases

RDS

- Aurora
- PostgeSQL
- MySQL
- MariaDB
- Oracle
- MS SQL Server

DynamoDB

DynamoDB

- NoSQL: a fully managed cloud database and supports both document and key-value store models
- Has tables, not databases
- Amazon DynamoDB Accelerator (DAX)-fully managed, highly available, in-memory cache

ElastiCache

ElastiCache

- In-memory data store and cache in the cloud
- Redis or Memcached
- Extreme performance and security

Redshift

Redshift

- Fast, simple, cost-effective data warehousing
- Standard SQL
- Redshift Spectrum: SQL exabytes of unstructured data in Amazon S3 without ETL

Redshift

Redshift creates a database when you provision a cluster. This database is used to load data and run queries on your data

Examples of how to load and query data:
http://tinyurl.com/ycghaqlp
http://docs.aws.amazon.com/redshift/latest/gsg/rs-gsg-create-sample-db.html

Module 5: PaaS

Working with ElasticBeanstalk

ElasticBeanstalk Benefits

- Easy and simple to get started
- Increased developer productivity
- Automatic scaling
- Allows for a complete resource control
- No additional charge for AWS Elastic Beanstalk

App Environments

- Python
- Ruby
- PHP
- Node.js
- Docker
- Ruby
- Java

Interfacting with ElasticBeanstalk

Develop Intelligence

- GitHub
- zip
- Docker
- AWS CLI (EB CLI is deprecated)
- IDEs
- WAR files

ElasticBeanstalk Resources

Develop Intelligence

Use Elastic Beanstalk to deploy a web app which uses RDS:

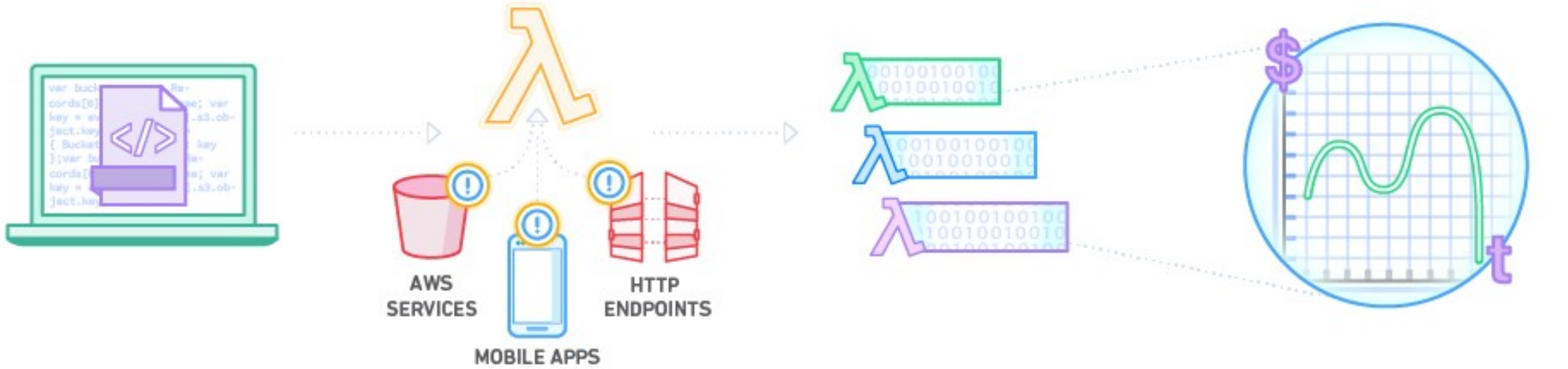
- http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_nodejs.html
- Getting Started with ElasticBeanstalk
- Developer Resources

Module 6: Serverless

Serverless with AWS Lambda

Benefits of AWS Lambda

- No Servers to Manage
- Continuous Scaling
- Subsecond Metering



Upload your code to AWS Lambda

Set up your code to trigger from other AWS services, HTTP endpoints, or in-app activity

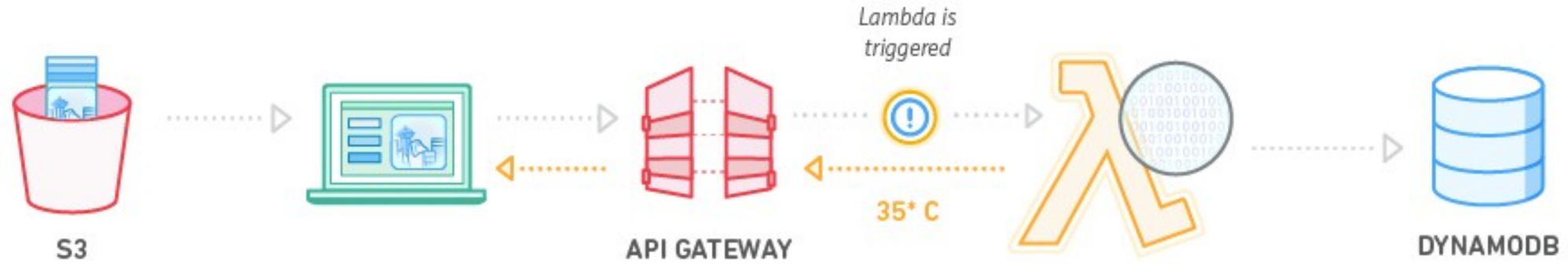
Lambda runs your code only when triggered, using only the compute resources needed

Pay just for the compute time you use

Use cases

- REST API and mobile backend
- IoT backend (Kinesis)
- Data ETL (Redshift, S3)
- Real-time file and stream processing
- Web applications (S3)

Example: *Weather Application*



Resources

- http://docs.aws.amazon.com/lambda/latest/dg/with-ddb.html
- https://github.com/dwyl/learn-aws-lambda#hello-world-example-inline

Lambda and DynamoDB

GET https://h8uwddrasb.execute-api.us-west-1.amazonaws.com/prod/my-first-fn?TableName=my-first-table

POST https://h8uwddrasb.execute-api.us-west-1.amazonaws.com/prod/my-first-fn?TableName=my-first-table

```
{ "TableName":    "myfirst-table",

"Item": {
"main-part":"1",
   "username":"CJA402",
   "password":  "cldjmPr!01"
}}
```

Develop Intelligence

Lab 5: Create a microservice to save data in DB

Questions?

Develop Intelligence

The end!

Many thanks to Azat Mardan @azat_co, Node University and DevelopIntelligence

Node University