Infrastructure as Code

Infrastructure as Code

Doing the same thing over and over again

- So far what we’ve done in AWS has been done “by hand”
- This is fine for development and experimentation
- Once you have things figured out however, you want to codify your infrastructure
  - AWS CLI
  - CloudFormation
  - Python SDK (`boto3`)
  - TerraForm

Infrastructure as Code

aws-cli

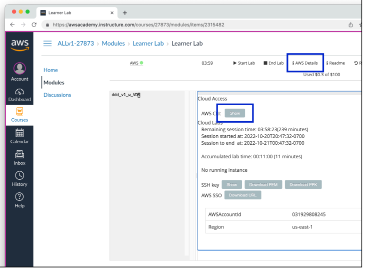
- On your EC2 instance, the AWS CLI is pre-installed
- You can install it on your laptop too
  - https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

Infrastructure as Code
aws-cli

- You need IAM credentials from your AWS account to use the CLI
- Log in to AWS Academy
  - https://awsacademy.instructure.com/login/canvas
- Start your AWS environment

Infrastructure as Code
aws-cli

- Under AWS Details
- Click on the “Show” button for AWS CLI



Cloud Access

AWS CLI: Show

Cloud Labs

Remaining session time: 03:58:23(239 minutes)
Session started at: 2022-10-20T20:47:32-0700
Session to end at: 2022-10-21T00:47:32-0700

Accumulated lab time: 00:11:00 (11 minutes)

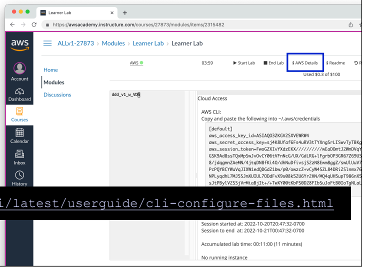
No running instance

SSH key
AWS SSO

AWSAccountId	031929808245
Region	us-east-1

Infrastructure as Code
aws-cli

- Copy the contents of the expanded box in to a new file in your user’s home directory, inside the hidden `~/.aws/` folder named `credentials`.
- See lecture slides 07-aws for walkthrough of setting up credentials in VS Code



Cloud Access

AWS CLI:
Copy and paste the following into ~/.aws/credentials

```
[default]
aws_access_key_id=ASIAQO3ZKGV2SXVEWRW4
aws_secret_access_key=sj4K8Ufaf6Fs4uRV3tTYXngSrLISwvTyTBKg
aws_session_token=FwoGZXIvYXdzEKX//////////wEaDOmtJ2WmDVqY
GSK9Ad8ssTQxMpSwJvOvCY06tVFnNcG/UX/GdLRG+lFgrbOP3GR67Z69US
8/jdqgmnZAeMN/4jtqDN8fKi4O/dhNuDfivsjS2zN8Ewm8ggZ/swUlUuV7
PcPQYBCYWuVqJIXW1edQDGd21bw/p0/owzcZ+vCyW4SZL84DRi2Slnmx76
NPLyqdhL7MJ5SJmXUIUL7ODdFvX9s08k52U6Yr2HN/MQ4qUH5upT986nX5
sJtPBylV2SSjVrMiaBjIt+/+TwXY00tKbPS0D28FIbSuJoFtB0IoTgNLaL
```

https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html

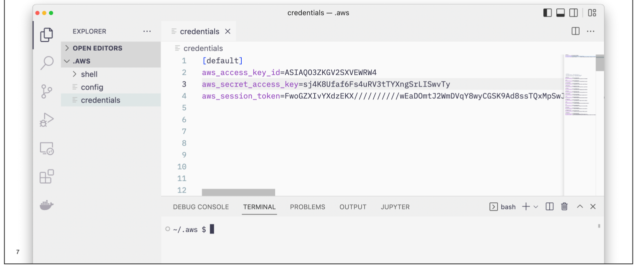
Session started at: 2022-10-20T20:47:32-0700
Session to end at: 2022-10-21T00:47:32-0700

Accumulated lab time: 00:11:00 (11 minutes)

No running instance

Infrastructure as Code

aws-cli



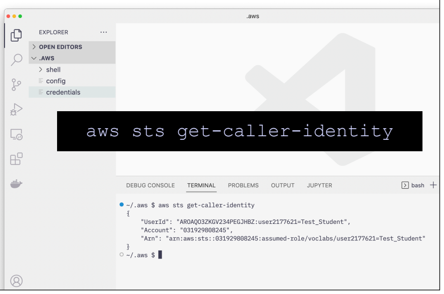
```
credentials
[default]
aws_access_key_id=ASIAQO3ZKGV2SXVEWRW4
aws_secret_access_key=sj4K8Ufaf6Fs4uRV3tTYXngSrLISwvTy
aws_session_token=FwoGZXIvYXdzEKX//////////wEaDOmtJ2WmDVqY8wyCGSK9Ad8ssTQxMpSwJ
```

```
~/.aws $ 
```

Infrastructure as Code

Who are you?

- Get some basic info about your credentials and make sure everything is working



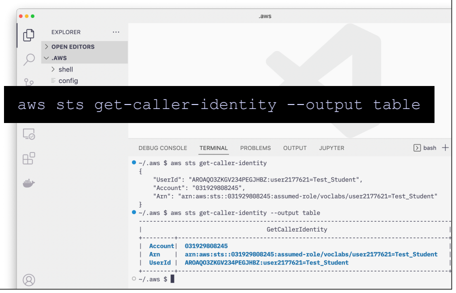
```
aws sts get-caller-identity
```

```
~/.aws $ aws sts get-caller-identity
{
    "UserId": "AROAQO3ZKGV234PEGJHBZ:user2177621=Test_Student",
    "Account": "031929808245",
    "Arn": "arn:aws:sts::031929808245:assumed-role/voclabs/user2177621=Test_Student"
}
~/.aws $ 
```

Infrastructure as Code

Who are you?

- Default output is JSON
- Can change to text or table



```
aws sts get-caller-identity --output table
```

```
~/.aws $ aws sts get-caller-identity
{
    "UserId": "AROAQO3ZKGV234PEGJHBZ:user2177621=Test_Student",
    "Account": "031929808245",
    "Arn": "arn:aws:sts::031929808245:assumed-role/voclabs/user2177621=Test_Student"
}
~/.aws $ aws sts get-caller-identity --output table
------------------------------------------------------------------------------------
|                                GetCallerIdentity                                 |
+---------+------------------------------------------------------------------------+
|  Account|  031929808245                                                          |
|  Arn    |  arn:aws:sts::031929808245:assumed-role/voclabs/user2177621=Test_Student |
|  UserId |  AROAQO3ZKGV234PEGJHBZ:user2177621=Test_Student                        |
+---------+------------------------------------------------------------------------+
~/.aws $ 
```

Infrastructure as Code

aws-cli

- The aws-cli is a command line interface to the core AWS API
- Everything you can do with the Web Console, you can do with the API and CLI

Infrastructure as Code

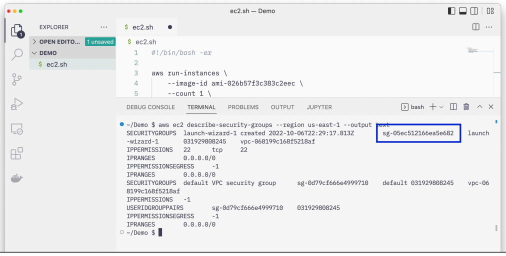
aws-cli

- If you've already created an EC2 instance, you have a security group already configured. Let's find it's ID

```
aws ec2 describe-security-groups --region us-east-1
```

Infrastructure as Code

aws-cli



```
#!/bin/bash -ex

aws run-instances \
    --image-id ami-026b57f3c383c2eec \
    --count 1 \
```

```
~/Demo $ aws ec2 describe-security-groups --region us-east-1 --output text
SECURITYGROUPS  launch-wizard-1 created 2022-10-06T22:29:17.813Z       sg-05ec512166ea5e682   launch
-wizard-1       031929808245    vpc-068199c168f5218af
IPPERMISSIONS   22      tcp     22
IPRANGES        0.0.0.0/0
IPPERMISSIONSEGRESS     -1
IPRANGES        0.0.0.0/0
SECURITYGROUPS  default VPC security group      sg-0d79cf666e4999710    default 031929808245    vpc-06
8199c168f5218af
IPPERMISSIONS   -1
USERIDGROUPPAIRS        sg-0d79cf666e4999710    031929808245
IPPERMISSIONSEGRESS     -1
IPRANGES        0.0.0.0/0
~/Demo $ 
```

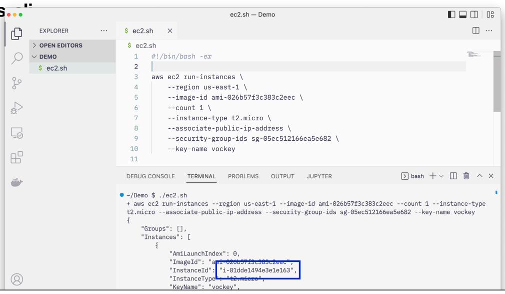
Infrastructure as Code

aws-cli

- Looking up information is fine, but can we make things?
- Let's deploy a new EC2 instance from the command line.

Infrastructure as Code

aws



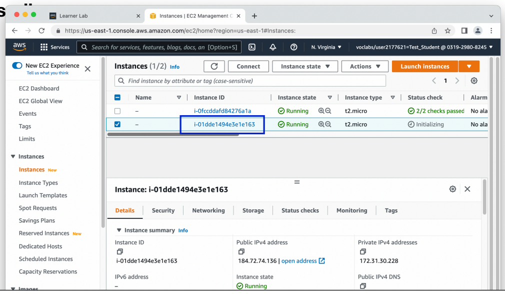
ec2.sh

```
#!/bin/bash -ex

aws ec2 run-instances \
    --region us-east-1 \
    --image-id ami-026b57f3c383c2eec \
    --count 1 \
    --instance-type t2.micro \
    --associate-public-ip-address \
    --security-group-ids sg-05ec512166ea5e682 \
    --key-name vockey
```

```
~/Demo $ ./ec2.sh
+ aws ec2 run-instances --region us-east-1 --image-id ami-026b57f3c383c2eec --count 1 --instance-type
t2.micro --associate-public-ip-address --security-group-ids sg-05ec512166ea5e682 --key-name vockey
{
    "Groups": [],
    "Instances": [
        {
            "AmiLaunchIndex": 0,
            "ImageId": "ami-026b57f3c383c2eec",
            "InstanceId": "i-01dde1494e3e1e163",
            "InstanceType": "t2.micro",
            "KeyName": "vockey",
```

Infrastructure as Code

aws



☐	Name	Instance ID	Instance state	Instance type	Status check	Alarm
☐	–	i-0fccddafd84276a1a	✓ Running	t2.micro	✓ 2/2 checks passed	No ala
☑	–	i-01dde1494e3e1e163	✓ Running	t2.micro	Initializing	No ala

Instance: i-01dde1494e3e1e163

Instance ID	Public IPv4 address	Private IPv4 addresses
i-01dde1494e3e1e163	184.72.74.136 | open address	172.31.30.228
IPv6 address	**Instance state**	**Public IPv4 DNS**
–	✓ Running	

CloudFormation

AWS CloudFormation

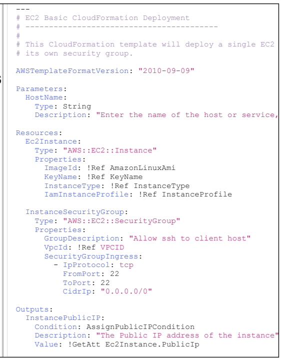
Amazon's first party Infrastructure as Code service

- Refers to both the templating syntax as well as the AWS service
- Create text file templates which can be repeatedly deployed
- A deployment is called a "stack"

AWS CloudFormation

Amazon's first party Infrastructure as

- Templates can be JSON or YAML formatted text files
- Top level sections: Parameters, Resources, Outputs and others
- Most data is basic key/value pairs
- YAML doesn't require you to quote every string

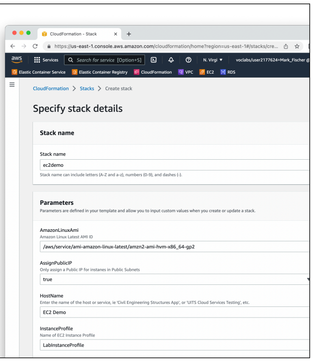


```
---
# EC2 Basic CloudFormation Deployment
# ------------------------------------------
#
# This CloudFormation template will deploy a single EC2
# its own security group.

AWSTemplateFormatVersion: "2010-09-09"

Parameters:
  HostName:
    Type: String
    Description: "Enter the name of the host or service,

Resources:
  Ec2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !Ref AmazonLinuxAmi
      KeyName: !Ref KeyName
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref InstanceProfile

  InstanceSecurityGroup:
    Type: "AWS::EC2::SecurityGroup"
    Properties:
      GroupDescription: "Allow ssh to client host"
      VpcId: !Ref VPCID
      SecurityGroupIngress:
        - IpProtocol: tcp
          FromPort: 22
          ToPort: 22
          CidrIp: "0.0.0.0/0"

Outputs:
  InstancePublicIP:
    Condition: AssignPublicIPCondition
    Description: "The Public IP address of the instance"
    Value: !GetAtt Ec2Instance.PublicIp
```

AWS CloudFormation

Infrastructure as Code service

- Templates can be uploaded to the AWS web console and deployed


CloudFormation - Stack
https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/cre...
aws
Services
Search for service [Option+S]
N. Virgi
voclabs/user2177624=Mark_Fischer @
Elastic Container Service
Elastic Container Registry
CloudFormation
VPC
EC2
RDS
CloudFormation > Stacks > Create stack

Specify stack details

Stack name

Stack name: ec2demo

Stack name can include letters (A-Z and a-z), numbers (0-9), and dashes (-).

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

AmazonLinuxAmi
Amazon Linux Latest AMI ID
/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2

AssignPublicIP
Only assign a Public IP for instanes in Public Subnets
true

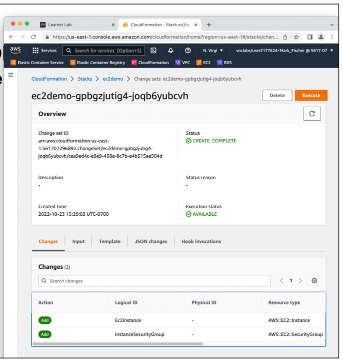
HostName
Enter the name of the host or service, ie 'Civil Engineering Structures App', or 'UITS Cloud Services Testing', etc.
EC2 Demo

InstanceProfile
Name of EC2 Instance Profile
LabInstanceProfile

AWS CloudFormatio

Infrastructure as Code service

- Stack changes can be previewed before deployment to see what resources will be created or modified


Learner Lab
CloudFormation - Stack ec2de
https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/chan...
aws
Services
Search for services [Option+S]
N. Virgi
voclabs/user2177624=Mark_Fischer @ 5617-07
Elastic Container Service
Elastic Container Registry
CloudFormation
VPC
EC2
RDS
CloudFormation > Stacks > ec2demo > Change sets: ec2demo-gpbgzjutig4-joqb6yubcvh
Delete
Execute
Changes
Input
Template
JSON changes
Hook invocations
Search changes
1

ec2demo-gpbgzjutig4-joqb6yubcvh

Overview

Change set ID
arn:aws:cloudformation:us-east-1:561707296892:changeSet/ec2demo-gpbgzjutig4-joqb6yubcvh/cea9ed4c-e9e9-438a-8c7b-e4b515aa504d

Status
CREATE_COMPLETE

Description
-

Status reason
-

Created time
2022-10-23 15:20:02 UTC-0700

Execution status
AVAILABLE

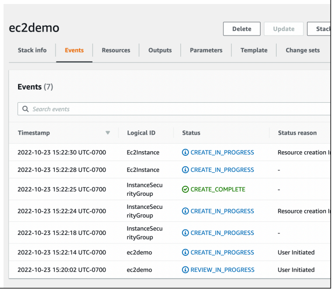
Changes (2)

Action	Logical ID	Physical ID	Resource type
Add	Ec2Instance	-	AWS::EC2::Instance
Add	InstanceSecurityGroup	-	AWS::EC2::SecurityGroup

AWS CloudFormation

Infrastructure as Code service

- Can watch the progress of the stack deployment
- If anything fails, CloudFormation can either leave things in place and broken so you can examine things, or it can roll back all your changes



ec2demo

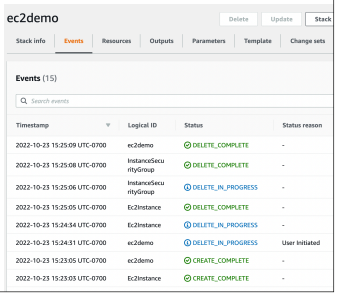
Events (7)

Timestamp	Logical ID	Status	Status reason
2022-10-23 15:22:30 UTC-0700	Ec2Instance	CREATE_IN_PROGRESS	Resource creation I
2022-10-23 15:22:28 UTC-0700	Ec2Instance	CREATE_IN_PROGRESS	-
2022-10-23 15:22:25 UTC-0700	InstanceSecurityGroup	CREATE_COMPLETE	-
2022-10-23 15:22:24 UTC-0700	InstanceSecurityGroup	CREATE_IN_PROGRESS	Resource creation I
2022-10-23 15:22:18 UTC-0700	InstanceSecurityGroup	CREATE_IN_PROGRESS	-
2022-10-23 15:22:14 UTC-0700	ec2demo	CREATE_IN_PROGRESS	User Initiated
2022-10-23 15:20:02 UTC-0700	ec2demo	REVIEW_IN_PROGRESS	User Initiated

AWS CloudFormation

Infrastructure as Code service

- Stacks can be updated over time
- Stacks can be completely deleted when you’re finished with it



Events (15)

Timestamp	Logical ID	Status	Status reason
2022-10-23 15:25:09 UTC-0700	ec2demo	DELETE_COMPLETE	-
2022-10-23 15:25:08 UTC-0700	InstanceSecurityGroup	DELETE_COMPLETE	-
2022-10-23 15:25:06 UTC-0700	InstanceSecurityGroup	DELETE_IN_PROGRESS	-
2022-10-23 15:25:05 UTC-0700	Ec2Instance	DELETE_COMPLETE	-
2022-10-23 15:24:34 UTC-0700	Ec2Instance	DELETE_IN_PROGRESS	-
2022-10-23 15:24:31 UTC-0700	ec2demo	DELETE_IN_PROGRESS	User Initiated
2022-10-23 15:23:05 UTC-0700	ec2demo	CREATE_COMPLETE	-
2022-10-23 15:23:03 UTC-0700	Ec2Instance	CREATE_COMPLETE	-

AWS Python SDK - boto3

AWS Language SDKs

Software Development Kit

- AWS Provides many ways to interact with its API
- RAW REST API
- AWS Web Console
- AWS CLI
- Programming Language SDKs

AWS Language SDKs

Programming Language SDKs

`https://aws.amazon.com/developer/tools/`

- Python
- JavaScript
- Node.js
- Java
- Go
- C++
- .NET
- Ruby
- Rust
- Swift

Python SDK - `boto3`

Authentication

- Just like the `aws-cli`, if you're making AWS API calls from outside of an AWS account, you need credentials
- The `boto3` SDK knows to look for your `[default]` credentials from your `~/.aws/credentials` file
- If you got the `aws-cli` working, then running python code from your laptop will also work
- If you want to run your python code inside of a container, you need to get credentials in to the container

Python SDK - `boto3`

Create an EC2 Instance

- The SDK documentation is essential

`https://boto3.amazonaws.com/v1/documentation/api/latest/index.html`

Python SDK - `boto3`

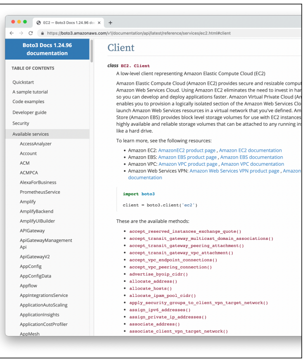
Two SDK Models

- Each Service in the boto3 library presents two different interface models
- `client` model
  - Closely maps directly to the AWS API itself / `aws-cli`
  - Returns dictionary mappings of the raw `JSON` responses
- `resource` model
  - More object oriented
  - Returns python objects

Python SDK - `boto3`

Create an EC2 Instance

- We want the `boto3.client` for EC2 to start
- Documentation provides a comprehensive list of all the properties and methods available
- Many examples
- I almost always start here first, then go off to more broad searches if I need to



Client

class `EC2. Client`

A low-level client representing Amazon Elastic Compute Cloud (EC2)

Amazon Elastic Compute Cloud (Amazon EC2) provides secure and resizable comput
Amazon Web Services Cloud. Using Amazon EC2 eliminates the need to invest in har
so you can develop and deploy applications faster. Amazon Virtual Private Cloud (Ar
enables you to provision a logically isolated section of the Amazon Web Services Clo
launch Amazon Web Services resources in a virtual network that you've defined. Am
Store (Amazon EBS) provides block level storage volumes for use with EC2 instances
highly available and reliable storage volumes that can be attached to any running ins
like a hard drive.

To learn more, see the following resources:

- Amazon EC2: AmazonEC2 product page , Amazon EC2 documentation
- Amazon EBS: Amazon EBS product page , Amazon EBS documentation
- Amazon VPC: Amazon VPC product page , Amazon VPC documentation
- Amazon Web Services VPN: Amazon Web Services VPN product page , Amazon documentation

```
import boto3

client = boto3.client('ec2')
```

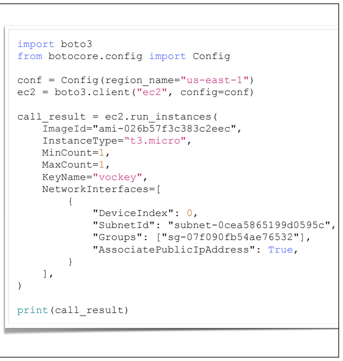
These are the available methods:

- `accept_reserved_instances_exchange_quote()`
- `accept_transit_gateway_multicast_domain_associations()`
- `accept_transit_gateway_peering_attachment()`
- `accept_transit_gateway_vpc_attachment()`
- `accept_vpc_endpoint_connections()`
- `accept_vpc_peering_connection()`
- `advertise_byoip_cidr()`
- `allocate_address()`
- `allocate_hosts()`
- `allocate_ipam_pool_cidr()`
- `apply_security_groups_to_client_vpn_target_network()`
- `assign_ipv6_addresses()`
- `assign_private_ip_addresses()`
- `associate_address()`
- `associate_client_vpn_target_network()`

Python SDK - `boto3`

Create an EC2 Instance

- Client version is `run_instances`
- Mostly matches the `aws-cli` but you can see similarities to the CloudFormation version as well
- Region is defined when creating the `client` object
- Requires more details for things like `NetworkInerfaces` and `Counts`

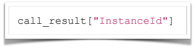


```
import boto3
from botocore.config import Config

conf = Config(region_name="us-east-1")
ec2 = boto3.client("ec2", config=conf)

call_result = ec2.run_instances(
    ImageId="ami-026b57f3c383c2eec",
    InstanceType=“t3.micro",
    MinCount=1,
    MaxCount=1,
    KeyName="vockey",
    NetworkInterfaces=[
        {
            "DeviceIndex": 0,
            "SubnetId": "subnet-0cea5865199d0595c",
            "Groups": ["sg-07f090fb54ae76532"],
            "AssociatePublicIpAddress": True,
        }
    ],
)

print(call_result)
```

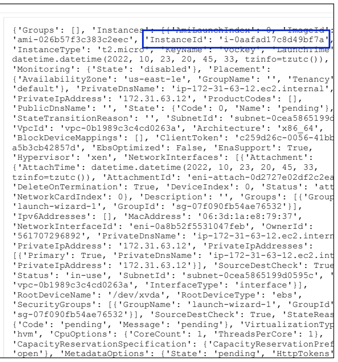
Python SDK - `boto3`

Create an EC2 Instance

- Response is a generic python dictionary with key/value pairs
- Useful if you only need cursory interaction with the resource after you create it



```
call_result["InstanceId"]
```

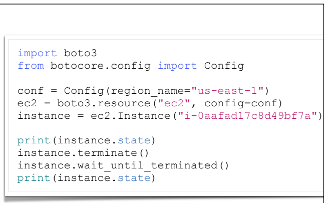


```
{'Groups': [], 'Instances': [{'AmiLaunchIndex': 0, 'ImageId':
'ami-026b57f3c383c2eec', 'InstanceId': 'i-0aafad17c8d49bf7a',
'InstanceType': 't2.micro', 'KeyName': 'vockey', 'LaunchTime'
datetime.datetime(2022, 10, 23, 20, 45, 33, tzinfo=tzutc()),
'Monitoring': {'State': 'disabled'}, 'Placement':
{'AvailabilityZone': 'us-east-1e', 'GroupName': '', 'Tenancy'
'default'}, 'PrivateDnsName': 'ip-172-31-63-12.ec2.internal',
'PrivateIpAddress': '172.31.63.12', 'ProductCodes': [],
'PublicDnsName': '', 'State': {'Code': 0, 'Name': 'pending'},
'StateTransitionReason': '', 'SubnetId': 'subnet-0cea5865199d
'VpcId': 'vpc-0b1989c3c4cd0263a', 'Architecture': 'x86_64',
'BlockDeviceMappings': [], 'ClientToken': 'c259d26c-0056-41bb
a5b3cb42857d', 'EbsOptimized': False, 'EnaSupport': True,
'Hypervisor': 'xen', 'NetworkInterfaces': [{'Attachment':
{'AttachTime': datetime.datetime(2022, 10, 23, 20, 45, 33,
tzinfo=tzutc()), 'AttachmentId': 'eni-attach-0d2727e02df2c2ea
'DeleteOnTermination': True, 'DeviceIndex': 0, 'Status': 'att
'NetworkCardIndex': 0}, 'Description': '', 'Groups': [{'Group
'launch-wizard-1', 'GroupId': 'sg-07f090fb54ae76532'}],
'Ipv6Addresses': [], 'MacAddress': '06:3d:1a:e8:79:37',
'NetworkInterfaceId': 'eni-0a8b52f5531047feb', 'OwnerId':
'561707296892', 'PrivateDnsName': 'ip-172-31-63-12.ec2.intern
'PrivateIpAddress': '172.31.63.12', 'PrivateIpAddresses':
[{'Primary': True, 'PrivateDnsName': 'ip-172-31-63-12.ec2.int
'PrivateIpAddress': '172.31.63.12'}], 'SourceDestCheck': True
'Status': 'in-use', 'SubnetId': 'subnet-0cea5865199d0595c', '
'vpc-0b1989c3c4cd0263a', 'InterfaceType': 'interface'}],
'RootDeviceName': '/dev/xvda', 'RootDeviceType': 'ebs',
'SecurityGroups': [{'GroupName': 'launch-wizard-1', 'GroupId'
'sg-07f090fb54ae76532'}], 'SourceDestCheck': True, 'StateReas
{'Code': 'pending', 'Message': 'pending'}, 'VirtualizationTyp
'hvm', 'CpuOptions': {'CoreCount': 1, 'ThreadsPerCore': 1},
'CapacityReservationSpecification': {'CapacityReservationPref
'open'}, 'MetadataOptions': {'State': 'pending', 'HttpTokens'
```

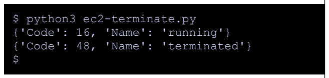
Python SDK - `boto3`

Terminate an EC2 Instance

- The resource model allows us to manipulate objects
- Here we first create an EC2 instance object in our code
- Because it is a python object, we can easily inspect attributes and call methods



```
import boto3
from botocore.config import Config

conf = Config(region_name="us-east-1")
ec2 = boto3.resource("ec2", config=conf)
instance = ec2.Instance("i-0aafad17c8d49bf7a")

print(instance.state)
instance.terminate()
instance.wait_until_terminated()
print(instance.state)
```



```
$ python3 ec2-terminate.py
{'Code': 16, 'Name': 'running'}
{'Code': 48, 'Name': 'terminated'}
$
```

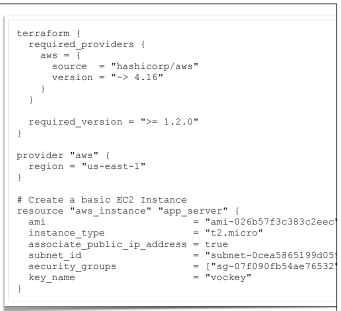
Terraform

Open-Source Multi-Provider Templating System

Terraform

Create an EC2 Instance

- Open-source tool spooned by HashiCorp
- Supports multiple cloud providers
- Has its own language that is similar to JSON, but supports comments, and built-in references and functions
- Install the `terraform` CLI tool



```
terraform {
  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 4.16"
    }
  }

  required_version = ">= 1.2.0"
}

provider "aws" {
  region = "us-east-1"
}

# Create a basic EC2 Instance
resource "aws_instance" "app_server" {
  ami                         = "ami-026b57f3c383c2eec
  instance_type               = "t2.micro"
  associate_public_ip_address = true
  subnet_id                   = "subnet-0cea5865199d05
  security_groups             = ["sg-07f090fb54ae76532
  key_name                    = "vockey"
}
```

https://www.terraform.io/downloads

Comparison

So what should you use?

- "It depends"
- Each method presented here has advantages and disadvantages
- Significant overlap between tools
- Can always start simple with a shell script running aws-cli commands. As that becomes cumbersome move to either boto3 or CloudFormation/Terraform depending on needs

Version Control Systems

Basically `git`

Version Control Systems

It's just `git` these days

- A version control system aims to keep track of all the changes made to any of your project files
- Mostly focused on text files
  - Binary files can be versioned, but they are harder to look at differences
- If you're dealing with text files that might change, you should probably use a version control system

Version Control Systems

It's just `git` these days

- Years ago there used to be several competing version control systems
- These days the industry has basically settled on `git`
- Originally developed to manage the Linux kernel.
- Designed as a distributed version control system with direct peer-to-peer capabilities
  - Very rarely used in practice
- Hub & spoke model of older version control systems gave rise to GitHub
- GitHub ≠ `git`!

The `git` Version Control System

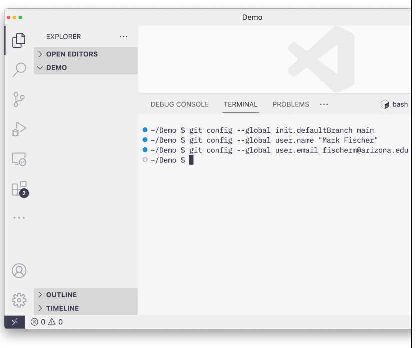
- A `git` repository is basically a folder with a hidden `.git` directory in it which contains state and history
- Files added to the folder can then be added to change sets and committed to the repository
- All of this can happen locally on your computer without needing a server
- If you want to use a service like GitHub, your local repository can be `pushed` to a remote repository hosted on GitHub.

`git` basics
Setup

- https://git-scm.com/downloads
- Many platforms have git installed by default
  - macOS has git as part of Xcode
  - Windows installer
  - Linux package managers

`git` basics
Setup

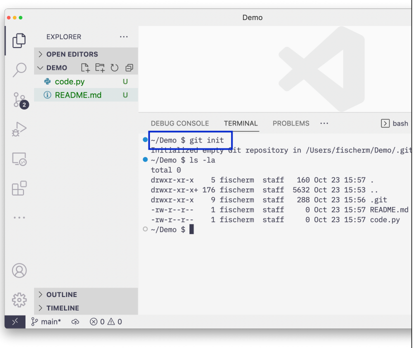
- Initial setup commands
- Set your default branch name
- Set your user.name
- Set your user.email



```
● ~/Demo $ git config --global init.defaultBranch main
● ~/Demo $ git config --global user.name "Mark Fischer"
● ~/Demo $ git config --global user.email fischerm@arizona.edu
○ ~/Demo $ █
```

`git` basics
Setup

- Create some files
- `git init` to initialize your current folder as a repository

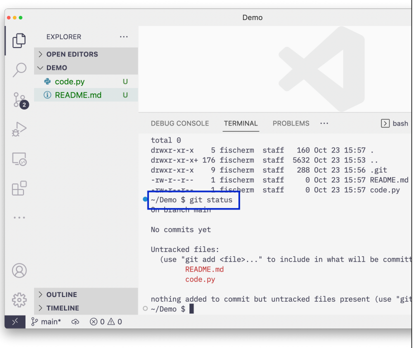


```
● ~/Demo $ git init
  Initialized empty Git repository in /Users/fischerm/Demo/.git
● ~/Demo $ ls -la
  total 0
  drwxr-xr-x    5 fischerm  staff   160 Oct 23 15:57 .
  drwxr-xr-x+ 176 fischerm  staff  5632 Oct 23 15:53 ..
  drwxr-xr-x    9 fischerm  staff   288 Oct 23 15:56 .git
  -rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 README.md
  -rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 code.py
○ ~/Demo $ █
```

`git` basics

Setup

- Use `git status` to show what changes are not in your repository

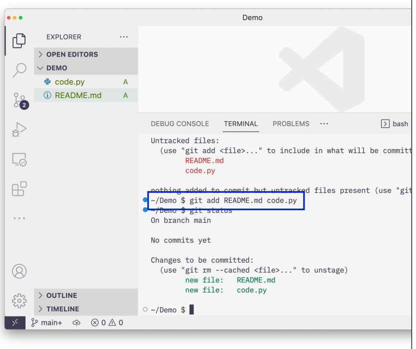


```
total 0
drwxr-xr-x    5 fischerm  staff   160 Oct 23 15:57 .
drwxr-xr-x+ 176 fischerm  staff  5632 Oct 23 15:53 ..
drwxr-xr-x    9 fischerm  staff   288 Oct 23 15:56 .git
-rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 README.md
-rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 code.py
~/Demo $ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committ
        README.md
        code.py

nothing added to commit but untracked files present (use "git
~/Demo $
```

`git` basics

Setup

- Use `git add` to stage new or changed files

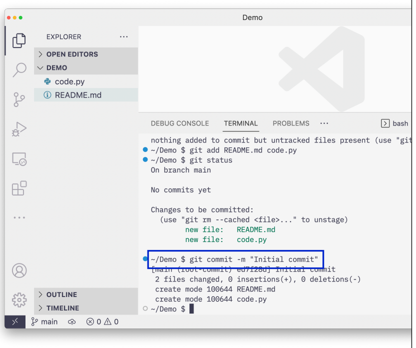


```
Untracked files:
  (use "git add <file>..." to include in what will be committ
        README.md
        code.py

nothing added to commit but untracked files present (use "git
~/Demo $ git add README.md code.py
~/Demo $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   code.py

~/Demo $
```

`git` basics

Setup

- Use `git commit` to commit all staged changes to the repository along with a change log message
- Message can be provided inline with the `-m` option, or with a CLI text editor like `vim`

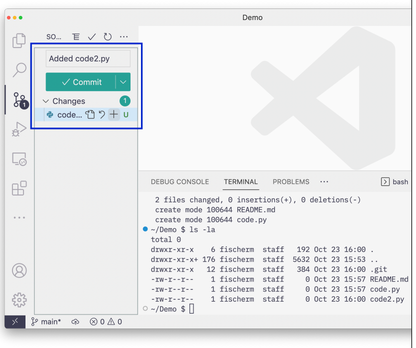


```
nothing added to commit but untracked files present (use "git
~/Demo $ git add README.md code.py
~/Demo $ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   README.md
        new file:   code.py

~/Demo $ git commit -m "Initial commit"
[main (root-commit) ed7f28d] Initial commit
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
 create mode 100644 code.py
~/Demo $
```

`git` basics

Setup

- Tools like VS Code have built-in support for `git`
- Add and commit changed files directly in VS Code GUI



Demo

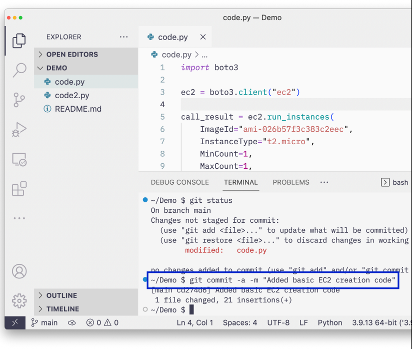
Added code2.py

```
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 README.md
 create mode 100644 code.py
~/Demo $ ls -la
total 0
drwxr-xr-x    6 fischerm  staff   192 Oct 23 16:00 .
drwxr-xr-x+ 176 fischerm  staff  5632 Oct 23 15:53 ..
drwxr-xr-x   12 fischerm  staff   384 Oct 23 16:00 .git
-rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 README.md
-rw-r--r--    1 fischerm  staff     0 Oct 23 15:57 code.py
-rw-r--r--    1 fischerm  staff     0 Oct 23 16:00 code2.py
~/Demo $ 
```

`git` basics

Setup

- Committing changes to files that are already tracked can be done with the `-a` option on the `git commit` command



code.py — Demo

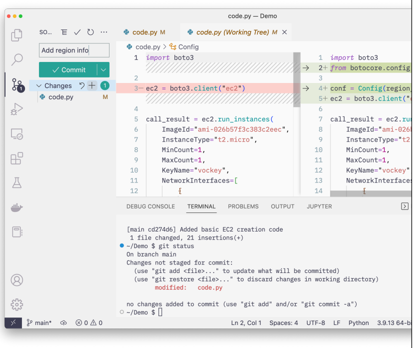
```
import boto3

ec2 = boto3.client("ec2")

call_result = ec2.run_instances(
    ImageId="ami-026b57f3c383c2eec",
    InstanceType="t2.micro",
    MinCount=1,
    MaxCount=1,
```

```
~/Demo $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working
        modified:   code.py

no changes added to commit (use "git add" and/or "git commit
~/Demo $ git commit -a -m "Added basic EC2 creation code"
[main cd274d6] Added basic EC2 creation code
 1 file changed, 21 insertions(+)
~/Demo $ 
```

`git` basics

Setup

- VS Code also has built-in support for showing differences between files as you work



code.py — Demo

Add region info

```
import boto3

ec2 = boto3.client("ec2")

call_result = ec2.run_instances(
    ImageId="ami-026b57f3c383c2eec",
    InstanceType="t2.micro",
    MinCount=1,
    MaxCount=1,
    KeyName="vockey",
    NetworkInterfaces=[
        {
```

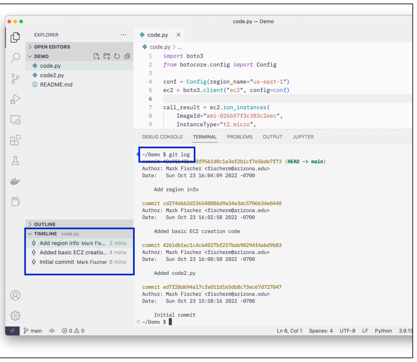
```
import boto3
from botocore.config
 
conf = Config(region
ec2 = boto3.client("
 
call_result = ec2.ru
    ImageId="ami-026
    InstanceType="t2
    MinCount=1,
    MaxCount=1,
    KeyName="vockey"
    NetworkInterface
        {
```

```
[main cd274d6] Added basic EC2 creation code
 1 file changed, 21 insertions(+)
~/Demo $ git status
On branch main
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   code.py

no changes added to commit (use "git add" and/or "git commit -a")
~/Demo $ 
```

`git` basics

Setup

- Can see a history of commits with the `git log` command
- Also shows up in the VS Code Timeline pane



Timeline (code.py):

- Add region info Mark Fis... 2 mins
- Added basic EC2 creatio... 4 mins
- Initial commit Mark Fischer 8 mins

code.py > ...

```python
import boto3
from botocore.config import Config

conf = Config(region_name="us-east-1")
ec2 = boto3.client("ec2", config=conf)

call_result = ec2.run_instances(
    ImageId="ami-026b57f3c383c2eec",
    InstanceType="t2.micro",
```

```
~/Demo $ git log
commit 40e91b48be[illegible]0f9561d0c1e3ef2b1cf7e56de7f73 (HEAD -> main)
Author: Mark Fischer <fischerm@arizona.edu>
Date:   Sun Oct 23 16:04:09 2022 -0700

    Add region info

commit cd274d662d236540086d9a34e3dc5706634e0440
Author: Mark Fischer <fischerm@arizona.edu>
Date:   Sun Oct 23 16:02:58 2022 -0700

    Added basic EC2 creation code

commit 4261db1ec1c4c64027bf257bab90294f4abd9b83
Author: Mark Fischer <fischerm@arizona.edu>
Date:   Sun Oct 23 16:00:58 2022 -0700

    Added code2.py

commit ed7f28d694a17cfe011d165db8c73ec67d727047
Author: Mark Fischer <fischerm@arizona.edu>
Date:   Sun Oct 23 15:58:16 2022 -0700

    Initial commit
~/Demo $
```