

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

The screenshot shows a web browser window with the URL `https://awsacademy.instructure.com/courses/27873/modules/items/2315482`. The page is titled "Learner Lab" and displays navigation links for "Home", "Modules", and "Discussions". A sidebar on the left contains navigation options: Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main content area shows a terminal window with the prompt `ddd_v1_w_VDS`. To the right, the "AWS Details" section is visible, featuring a "Show" button for the "AWS CLI" section, which is highlighted with a blue box. Other details include session time, accumulated lab time, and a table of AWS account information.

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

The screenshot shows the AWS Academy Learner Lab interface. The browser address bar displays `https://awsacademy.instructure.com/courses/27873/modules/items/2315482`. The page title is "ALLv1-27873 > Modules > Learner Lab > Learner Lab". The interface includes a sidebar with navigation options: Account, Dashboard, Courses, Calendar, Inbox, and History. The main content area shows a terminal window with the command `ddd_v1_w_VD$`. To the right, the "Cloud Access" section provides instructions for setting up AWS CLI credentials. A blue box highlights the "AWS Details" button, which shows "Used \$0.3 of \$100".

```
[default]
aws_access_key_id=ASIAQ03ZKGV2SXVEWRW4
aws_secret_access_key=sj4K8Ufaf6Fs4uRV3tTYXngSrLISwvTyTBKg
aws_session_token=FwoGZXIvYXdzEKX////////wEaD0mtJ2WmDVqY
GSK9Ad8ssTQxMpSwJv0vCY06tVFnNcG/UX/GdLRG+lFgrbOP3GR67Z69US
8/jdqgmZAE MN/4jtdN8fKi40/dhNuDfivsjs2zN8Ewm8ggZ/swU1UuV7
PcPQYBCYWuVqJIXW1edQDgD21bw/p0/owzcZ+vCyW4SZL84DRi2Slnmx76
NPLyqdhL7MJ5SJmXUIUL70DdFvX9s08k52U6Yr2HN/MQ4qUH5upT986nX5
sJtPBylV2SSjVrMiaBjIt+/+TwXY00tKbPS0D28FIBSuJoFtB0IoTgNLal
```

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

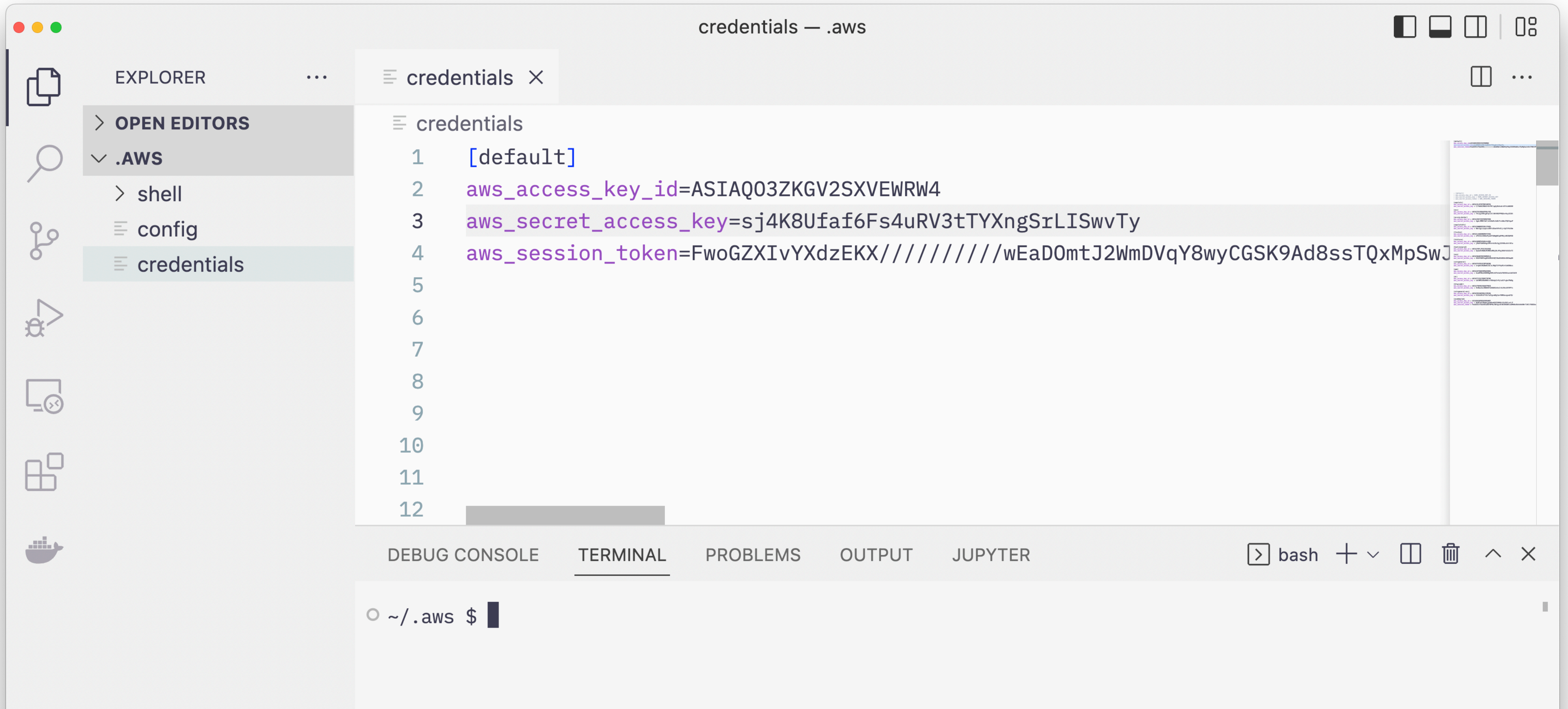
Accumulated lab time: 00:11:00 (11 minutes)

No running instance

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

Infrastructure as Code

aws-cli



The screenshot shows a code editor window titled "credentials — .aws". The Explorer sidebar on the left shows a project structure with a folder ".AWS" containing files "shell", "config", and "credentials". The "credentials" file is open in the editor, showing the following content:

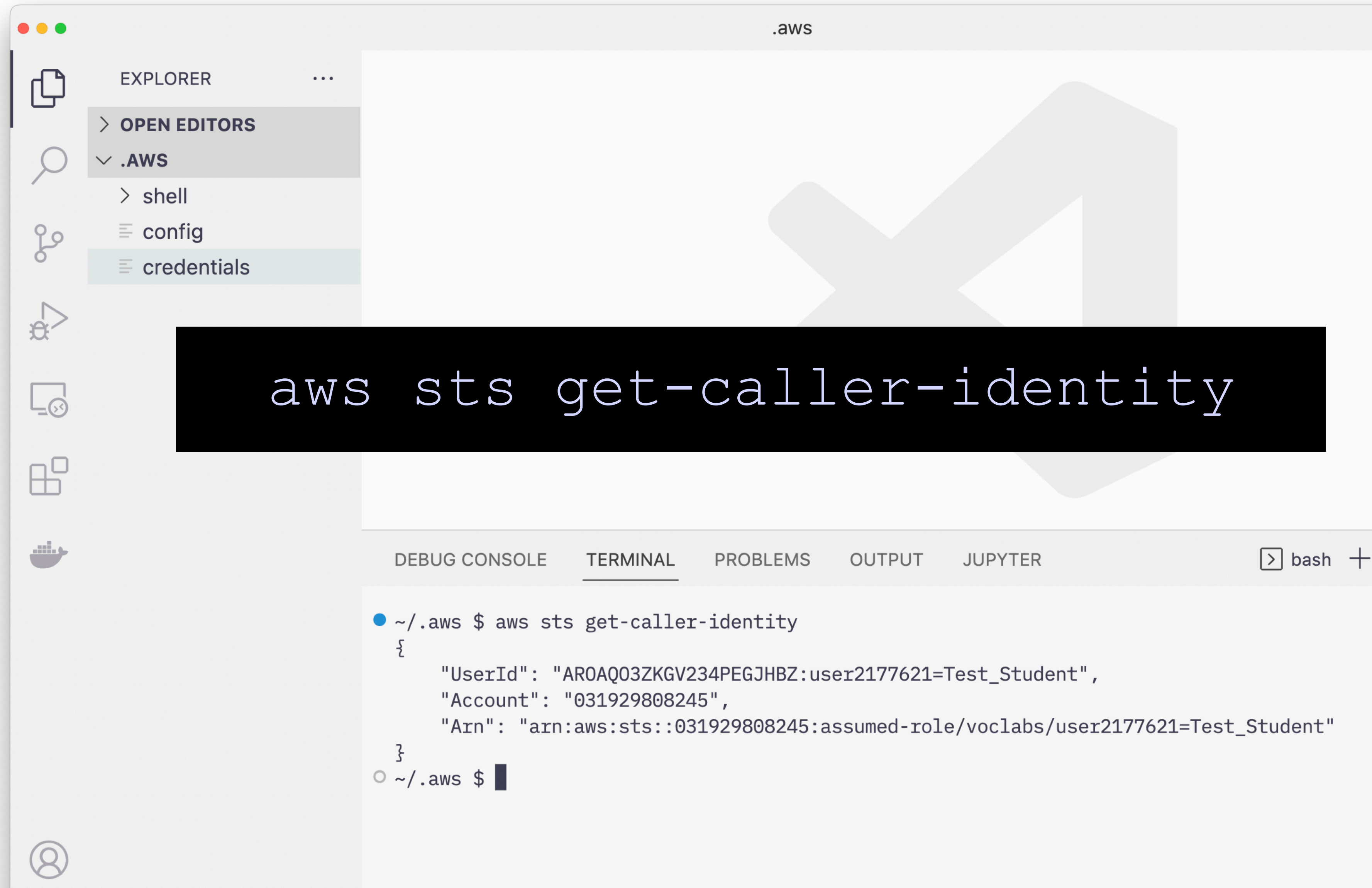
```
1 [default]
2 aws_access_key_id=ASIAQ03ZKGV2SXVEWRW4
3 aws_secret_access_key=sj4K8Ufaf6Fs4uRV3tTYXngSrLISwvTy
4 aws_session_token=FwoGZXIvYXdzEKX////////wEaD0mtJ2WmDVqY8wyCGSK9Ad8ssTQxMpSwJ
5
6
7
8
9
10
11
12
```

The editor interface includes a terminal at the bottom with the prompt "~/.aws \$". The terminal tab is labeled "bash".

Infrastructure as Code

Who are you?

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



The screenshot shows a code editor interface with a terminal window. The terminal window is titled ".aws" and shows the command `aws sts get-caller-identity` being executed. The output is a JSON object:

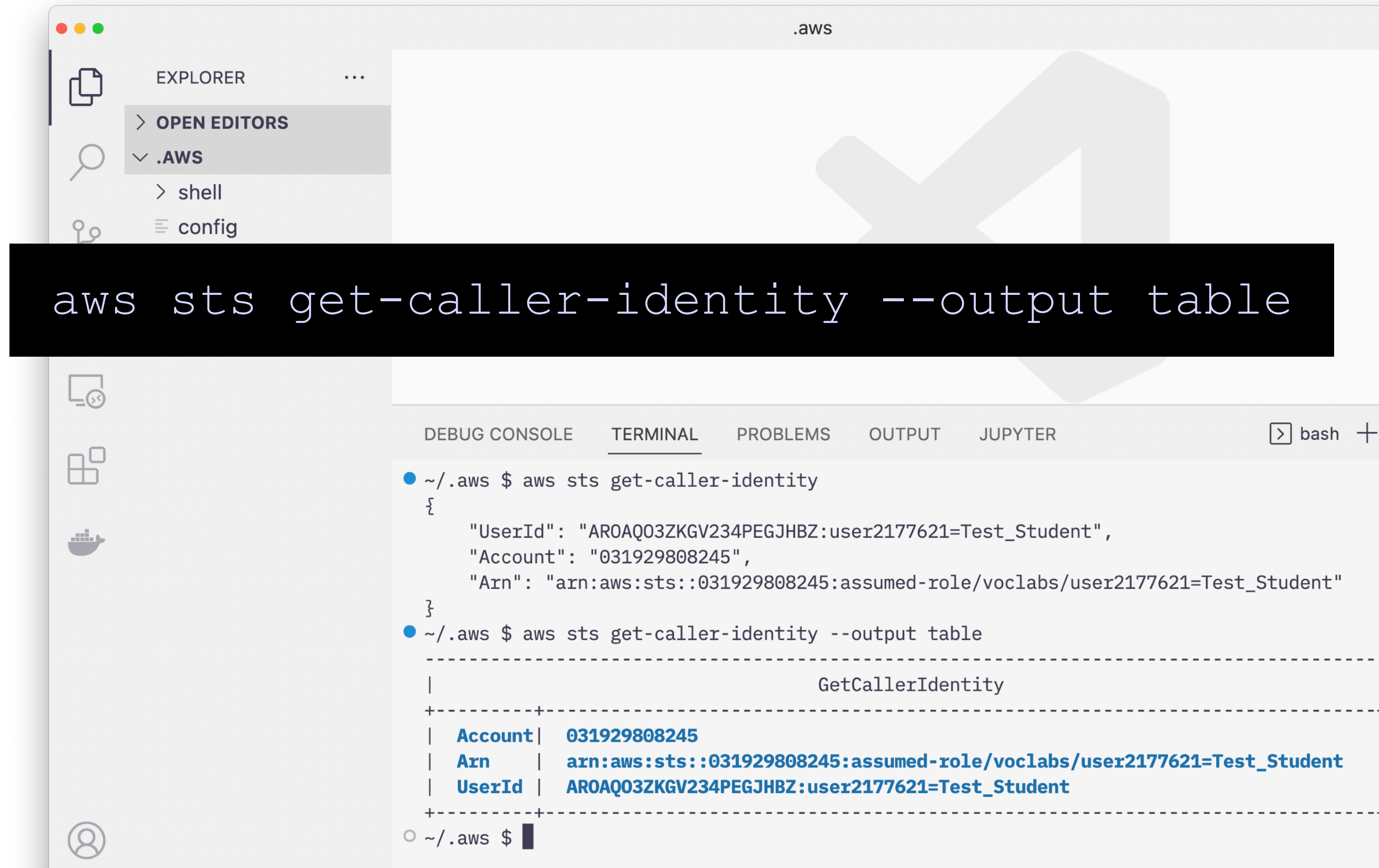
```
aws sts get-caller-identity
{
  "UserId": "AR0AQ03ZKGV234PEGJHBZ:user2177621=Test_Student",
  "Account": "031929808245",
  "Arn": "arn:aws:sts::031929808245:assumed-role/voclabs/user2177621=Test_Student"
}
```

The terminal window also shows the prompt `~/ .aws $` and a cursor. The code editor interface includes an Explorer sidebar on the left with a tree view showing the file structure: `EXPLORER`, `OPEN EDITORS`, `.AWS`, `shell`, `config`, and `credentials`. The terminal window has tabs for `DEBUG CONSOLE`, `TERMINAL`, `PROBLEMS`, `OUTPUT`, and `JUPYTER`. The terminal window title bar shows `bash` and a plus sign.

Infrastructure as Code

Who are you?

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



The screenshot shows a VS Code editor window with a terminal pane. The Explorer sidebar on the left shows a file tree with folders for 'EXPLORER', 'OPEN EDITORS', '.AWS', 'shell', and 'config'. The terminal pane is active and shows the following commands and output:

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

```
~/ .aws $ aws sts get-caller-identity
{
  "UserId": "AROAQ03ZKGV234PEGJHBZ: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  | AROAQ03ZKGV234PEGJHBZ: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



The screenshot shows a code editor window titled "ec2.sh — Demo". The Explorer sidebar on the left shows a file named "ec2.sh" under a "DEMO" folder. The editor displays the following shell script:

```
$ ec2.sh
1  #!/bin/bash -ex
2
3  aws run-instances \
4  --image-id ami-026b57f3c383c2eec \
5  --count 1 \
```

Below the script, the terminal output is shown. The command executed is `aws ec2 describe-security-groups --region us-east-1 --output text`. The output lists security groups, with the ID `sg-05ec512166ea5e682` highlighted in a blue box. The output also shows details for a "default VPC security group".

```
~/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

The image shows a code editor window titled "ec2.sh — Demo" with a file explorer on the left. The file explorer shows a folder named "DEMO" containing a file named "ec2.sh". The code editor displays the following script:

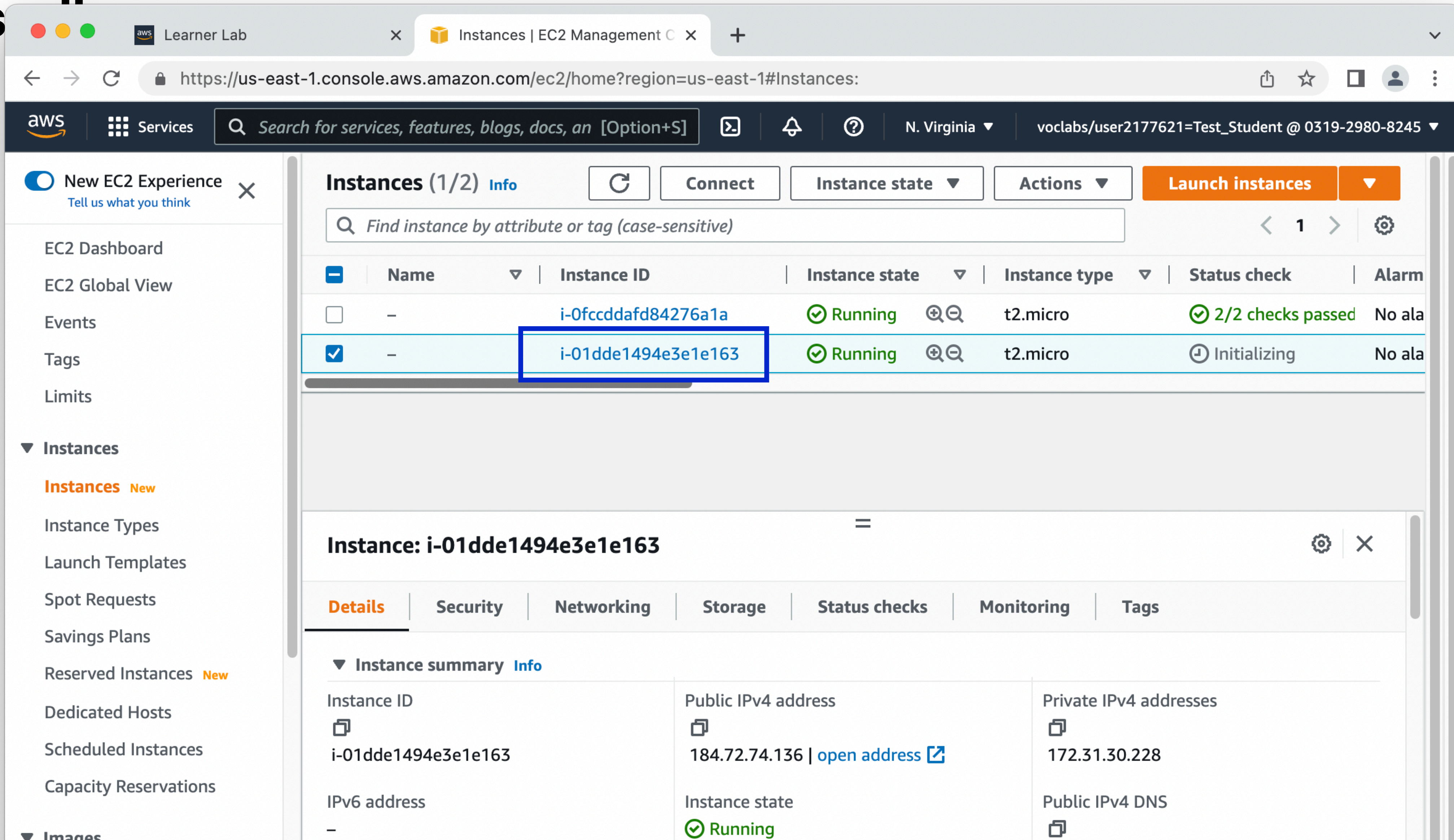
```
$ ec2.sh
$ ec2.sh
1  #!/bin/bash -ex
2
3  aws ec2 run-instances \
4    --region us-east-1 \
5    --image-id ami-026b57f3c383c2eec \
6    --count 1 \
7    --instance-type t2.micro \
8    --associate-public-ip-address \
9    --security-group-ids sg-05ec512166ea5e682 \
10   --key-name vockey
11
```

Below the code editor is a terminal window showing the execution of the script. The terminal output is as follows:

```
~/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



The screenshot displays the AWS Management Console interface for EC2 instances. The main content area shows a table of instances with the following data:

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

The instance **i-01dde1494e3e1e163** is highlighted with a blue box. Below the table, the details for this instance are shown:

Instance: i-01dde1494e3e1e163

- Details** | Security | Networking | Storage | Status checks | Monitoring | Tags
- Instance summary** Info
 - Instance ID: i-01dde1494e3e1e163
 - Public IPv4 address: 184.72.74.136 | [open address](#)
 - Private IPv4 addresses: 172.31.30.228
 - Instance state: Running
 - Public IPv4 DNS: [copy icon]

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,"

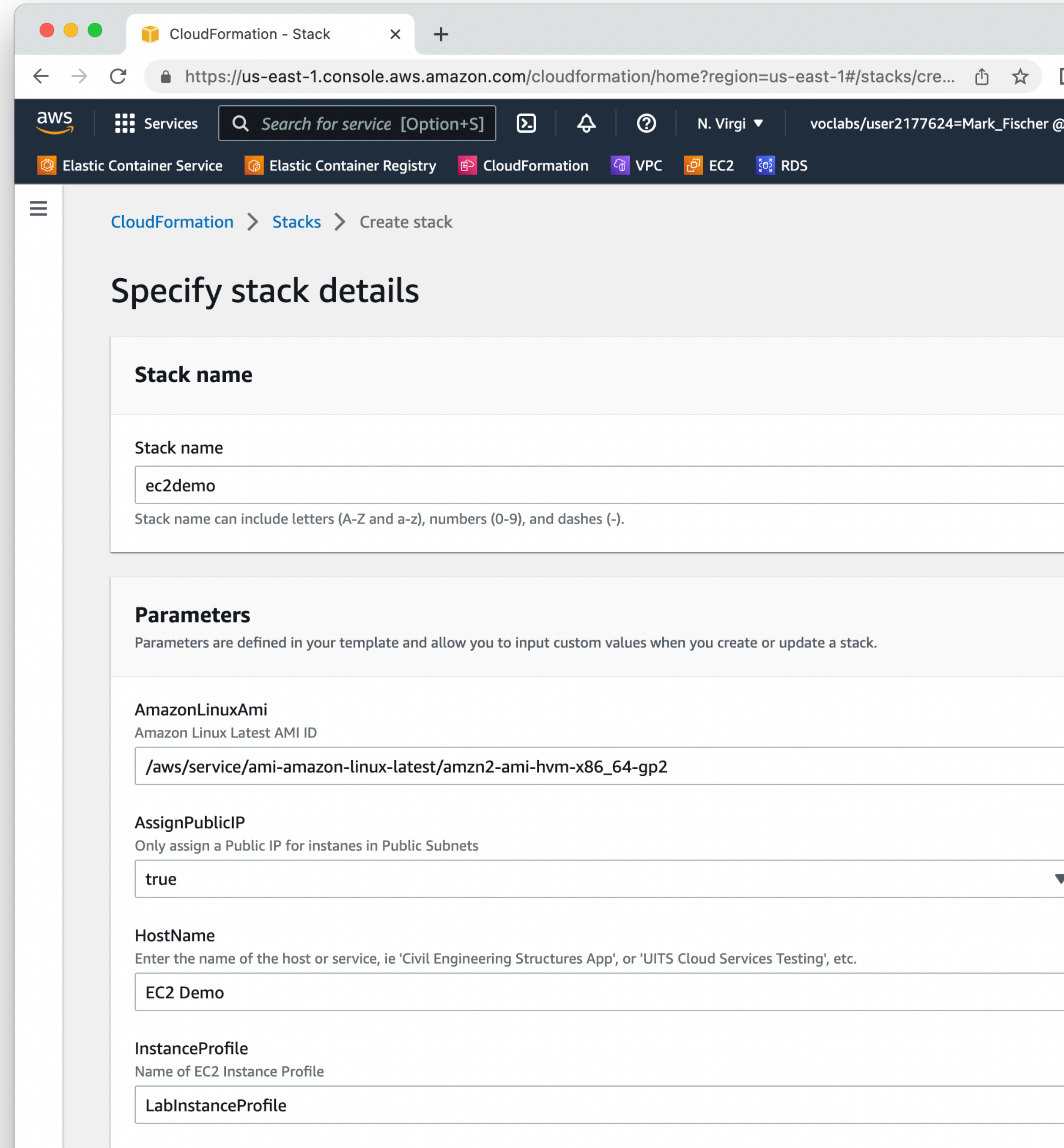
Resources:
  Ec2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !Ref AmazonLinuxAmi
      KeyName: !Ref KeyName
      InstanceType: !Ref InstanceType
      IamInstanceProfile: !Ref InstanceProfile

  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



AWS CloudFormation Infrastructure as Code service

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

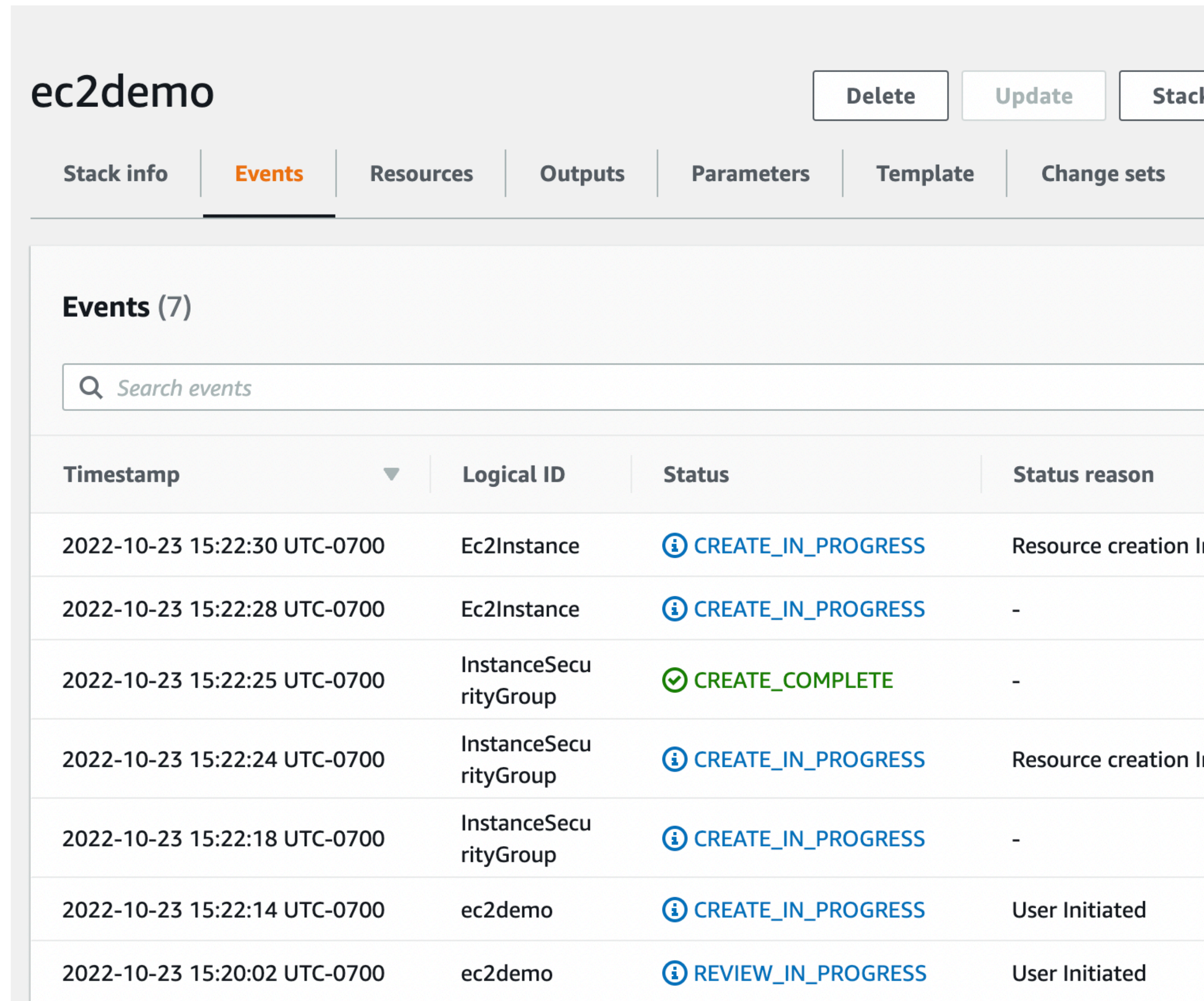
The screenshot displays the AWS CloudFormation console interface. The browser address bar shows the URL: `https://us-east-1.console.aws.amazon.com/cloudformation/home?region=us-east-1#/stacks/chan...`. The navigation bar includes the AWS logo, a search bar, and various service icons like Elastic Container Service, Elastic Container Registry, CloudFormation, VPC, EC2, and RDS. The main content area shows the details for a change set named `ec2demo-gpbgzjutig4-joqb6yubcvh`. The status is `CREATE_COMPLETE` and the execution status is `AVAILABLE`. Below the overview, there are tabs for `Changes`, `Input`, `Template`, `JSON changes`, and `Hook invocations`. The `Changes` tab is active, showing a table with two resources to be added:

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



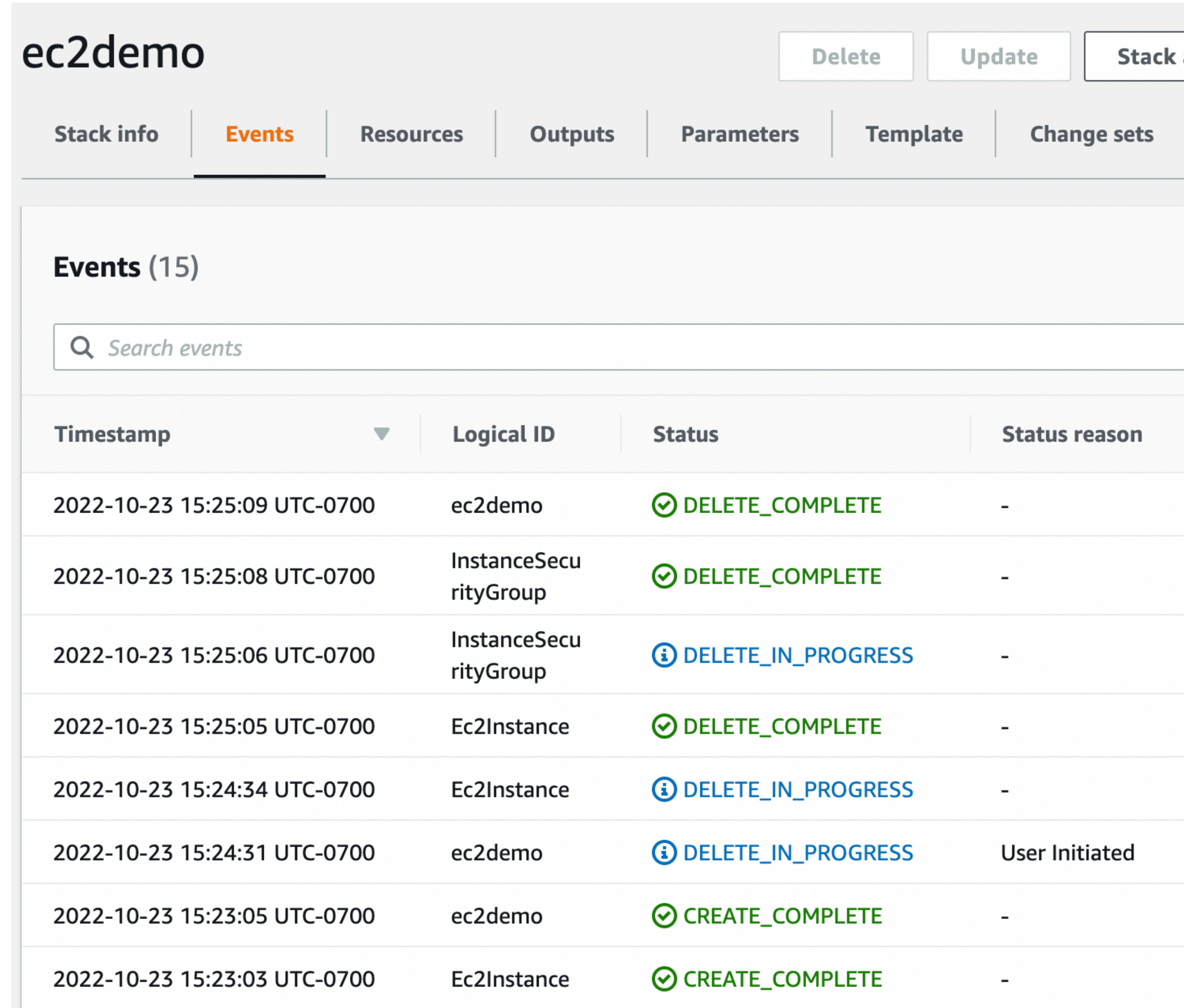
The screenshot shows the AWS CloudFormation console for a stack named 'ec2demo'. The 'Events' tab is selected, displaying a list of 7 events. The events table includes columns for Timestamp, Logical ID, Status, and Status reason. The events show the stack being initiated, followed by the creation of InstanceSecurityGroup resources, and the creation of Ec2Instance resources.

Timestamp	Logical ID	Status	Status reason
2022-10-23 15:22:30 UTC-0700	Ec2Instance	CREATE_IN_PROGRESS	Resource creation in progress
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 in progress
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



ec2demo

Delete Update Stack

Stack info **Events** Resources Outputs Parameters Template Change sets

Events (15)

Q Search events

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

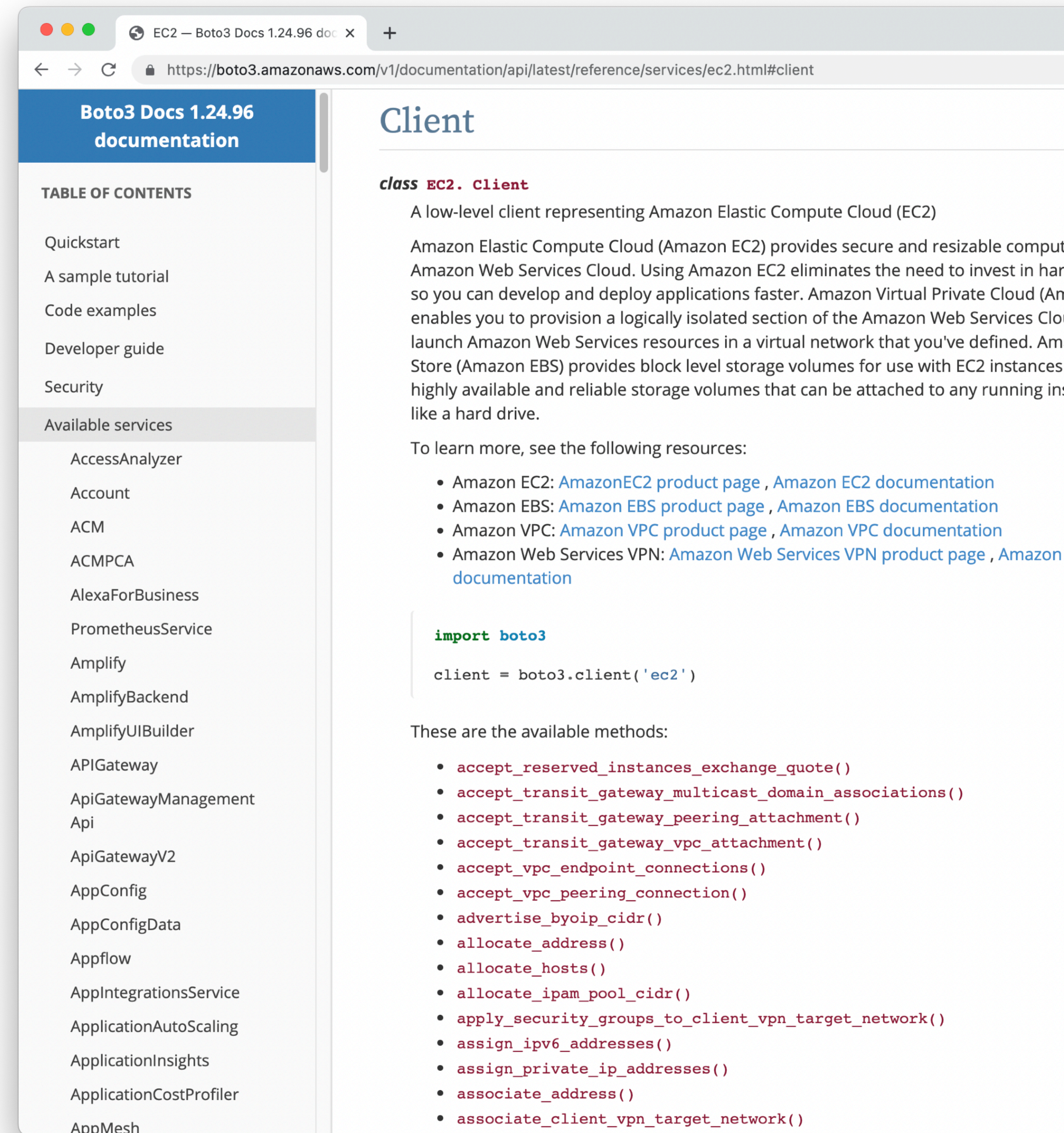
Two SDK Models

- 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



The screenshot shows a web browser displaying the Boto3 documentation for the EC2 client. The page title is "Client" and the URL is `https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html#client`. The left sidebar contains a "TABLE OF CONTENTS" with links to Quickstart, A sample tutorial, Code examples, Developer guide, Security, and Available services. The "Available services" list includes AccessAnalyzer, Account, ACM, ACMPCA, AlexaForBusiness, PrometheusService, Amplify, AmplifyBackend, AmplifyUIBuilder, APIGateway, ApiGatewayManagementApi, ApiGatewayV2, AppConfig, AppConfigData, Appflow, AppIntegrationsService, ApplicationAutoScaling, ApplicationInsights, ApplicationCostProfiler, and AppMesh. The main content area describes the `class EC2. Client` as a low-level client representing Amazon Elastic Compute Cloud (EC2). It provides a detailed description of Amazon EC2 and lists resources for learning more, including links to Amazon EC2, Amazon EBS, Amazon VPC, and Amazon Web Services VPN product pages and documentation. A code block shows the import and client creation:

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

 Below the code, it lists the available methods for the client, such as `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()`, and `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 `NetworkInterfaces` 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-0cea5865199d0595c', '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': 'attached', 'NetworkCardIndex': 0}, 'Description': '', 'Groups': [{'Group': {'Name': '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.internal', 'PrivateIpAddress': '172.31.63.12', 'PrivateIpAddresses': [{'Primary': True, 'PrivateDnsName': 'ip-172-31-63-12.ec2.internal', 'PrivateIpAddress': '172.31.63.12'}], 'SourceDestCheck': True, 'Status': 'in-use', 'SubnetId': 'subnet-0cea5865199d0595c', 'VpcId': 'vpc-0b1989c3c4cd0263a', 'InterfaceType': 'interface'}], 'RootDeviceName': '/dev/xvda', 'RootDeviceType': 'ebs', 'SecurityGroups': [{'GroupName': 'launch-wizard-1', 'GroupId': 'sg-07f090fb54ae76532'}], 'SourceDestCheck': True, 'StateReason': {'Code': 'pending', 'Message': 'pending'}, 'VirtualizationType': 'hvm', 'CpuOptions': {'CoreCount': 1, 'ThreadsPerCore': 1}, 'CapacityReservationSpecification': {'CapacityReservationPreference': 'open'}, 'MetadataOptions': {'State': 'pending', 'HttpTokens': 'required'}}
```

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-0cea5865199d059"
  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 \neq `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

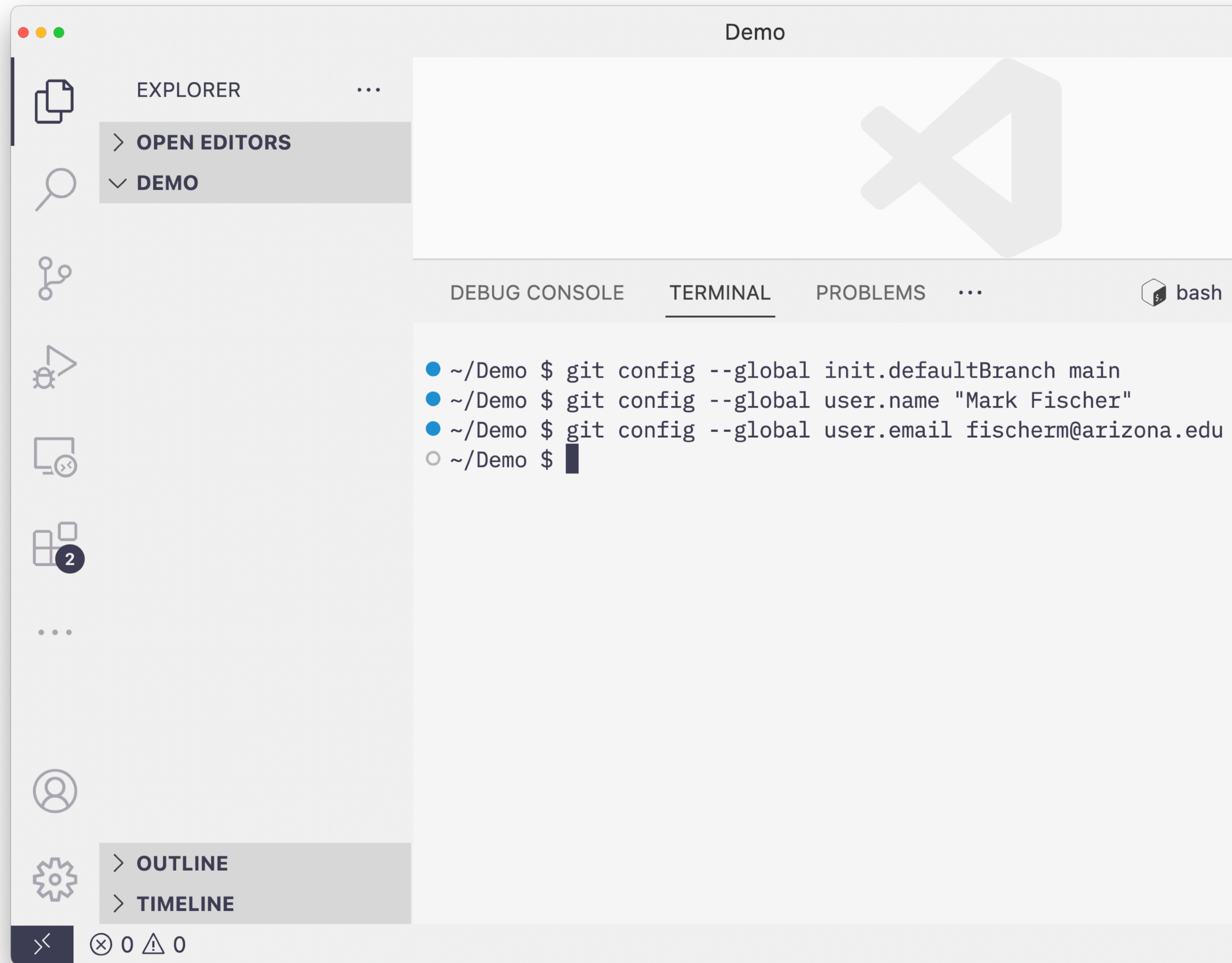
Setup

- <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



The screenshot shows the Visual Studio Code interface with a terminal window open. The terminal displays the following commands and their output:

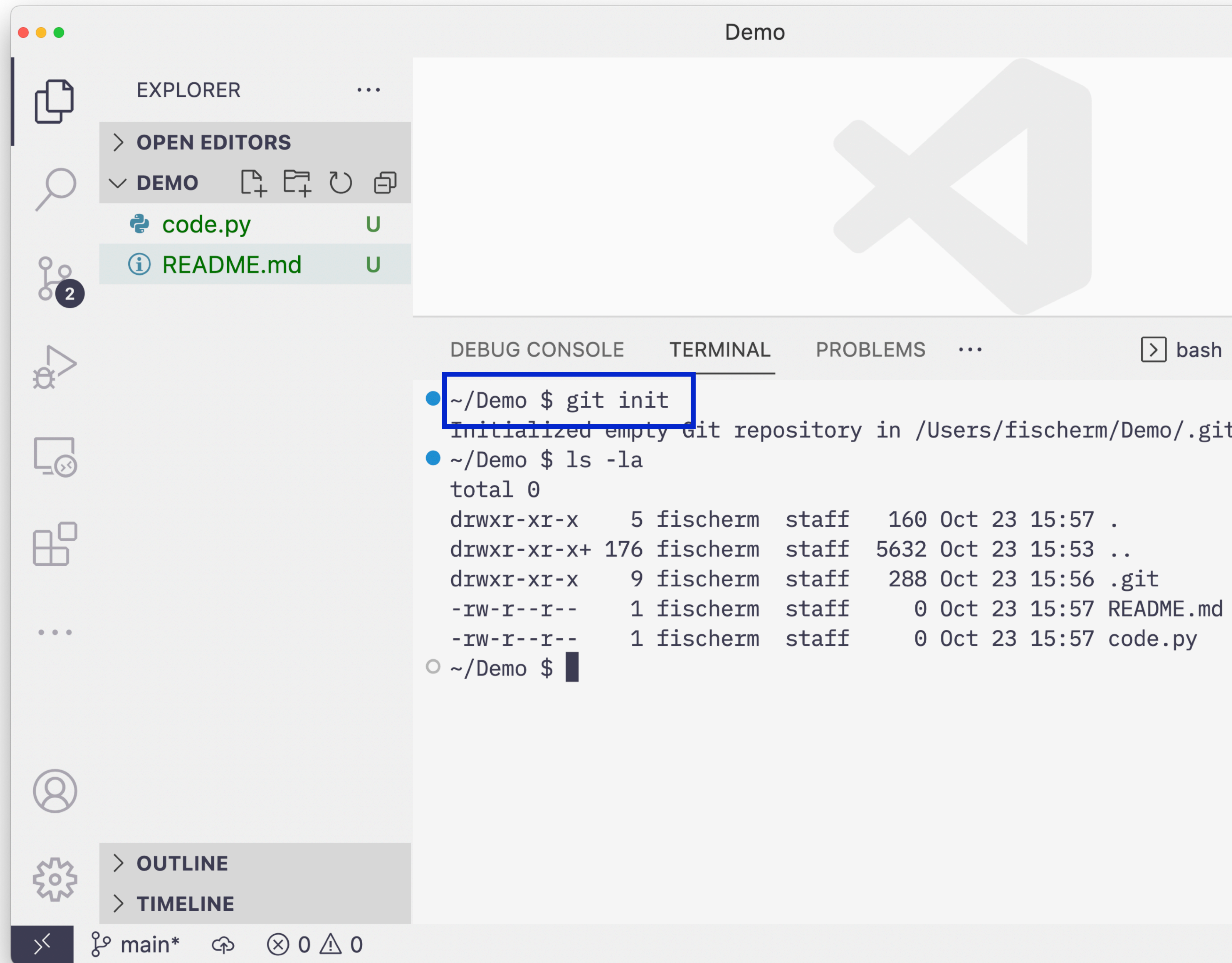
```
~/Demo $ git config --global init.defaultBranch main
~/Demo $ git config --global user.name "Mark Fischer"
~/Demo $ git config --global user.email fischerm@arizona.edu
~/Demo $
```

The interface includes a sidebar with icons for Explorer, Search, Source Control, Run and Debug, Remote Explorer, Extensions, and Settings. The main editor area shows the 'Demo' folder with a large 'X' icon. The terminal window is titled 'bash' and shows the current directory as '~/Demo'.

git basics

Setup

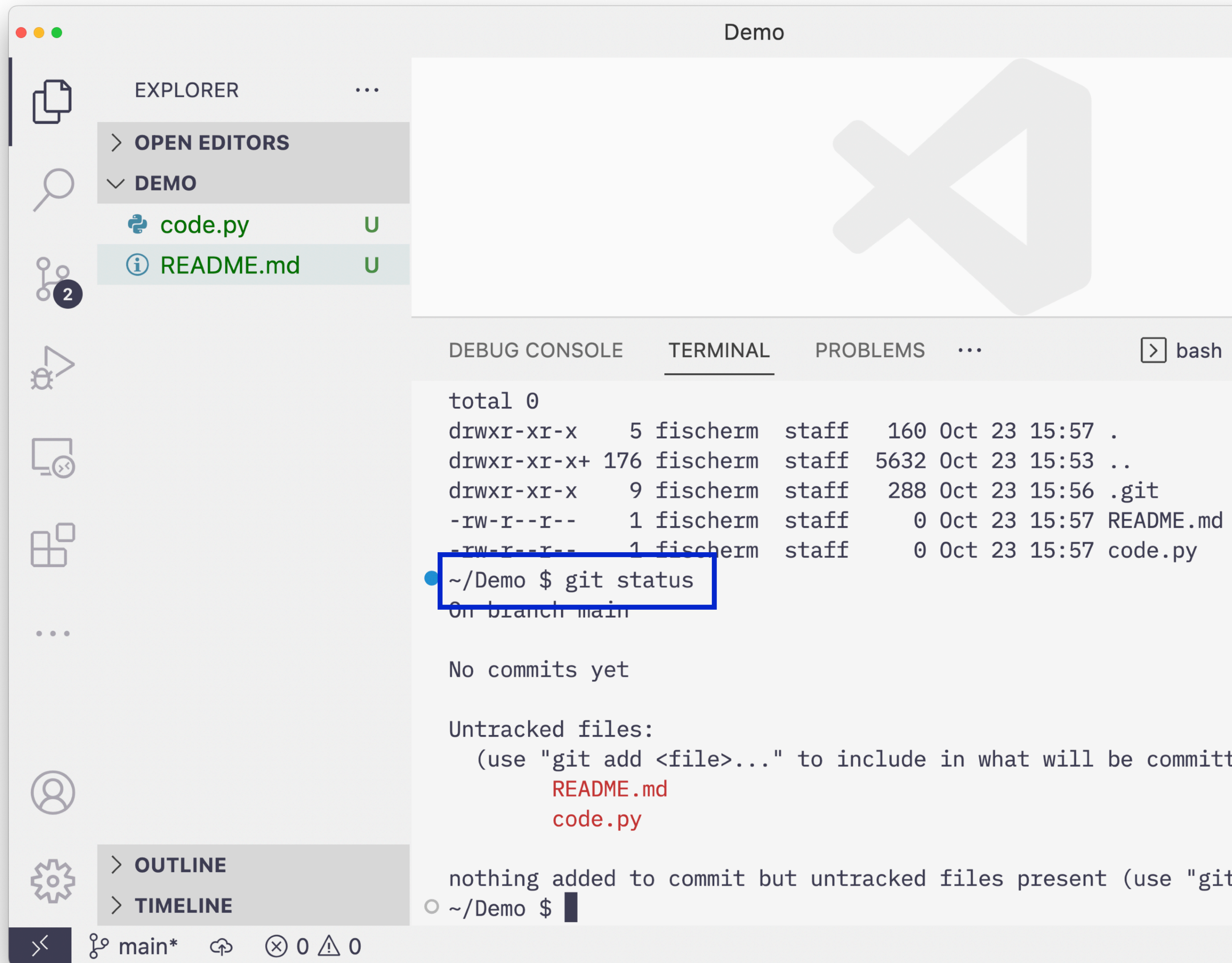
- Create some files
- `git init` to initialize your current folder as a repository



git basics

Setup

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



The screenshot shows the Visual Studio Code interface with a terminal window open. The Explorer sidebar on the left shows a project named 'DEMO' with two files: 'code.py' and 'README.md', both marked as 'U' (Untracked). The terminal window, titled 'Demo', shows the output of the `git status` command. The output indicates that there are no commits yet and that there are untracked files: 'README.md' and 'code.py'. The command prompt is `~/Demo $`.

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

No commits yet

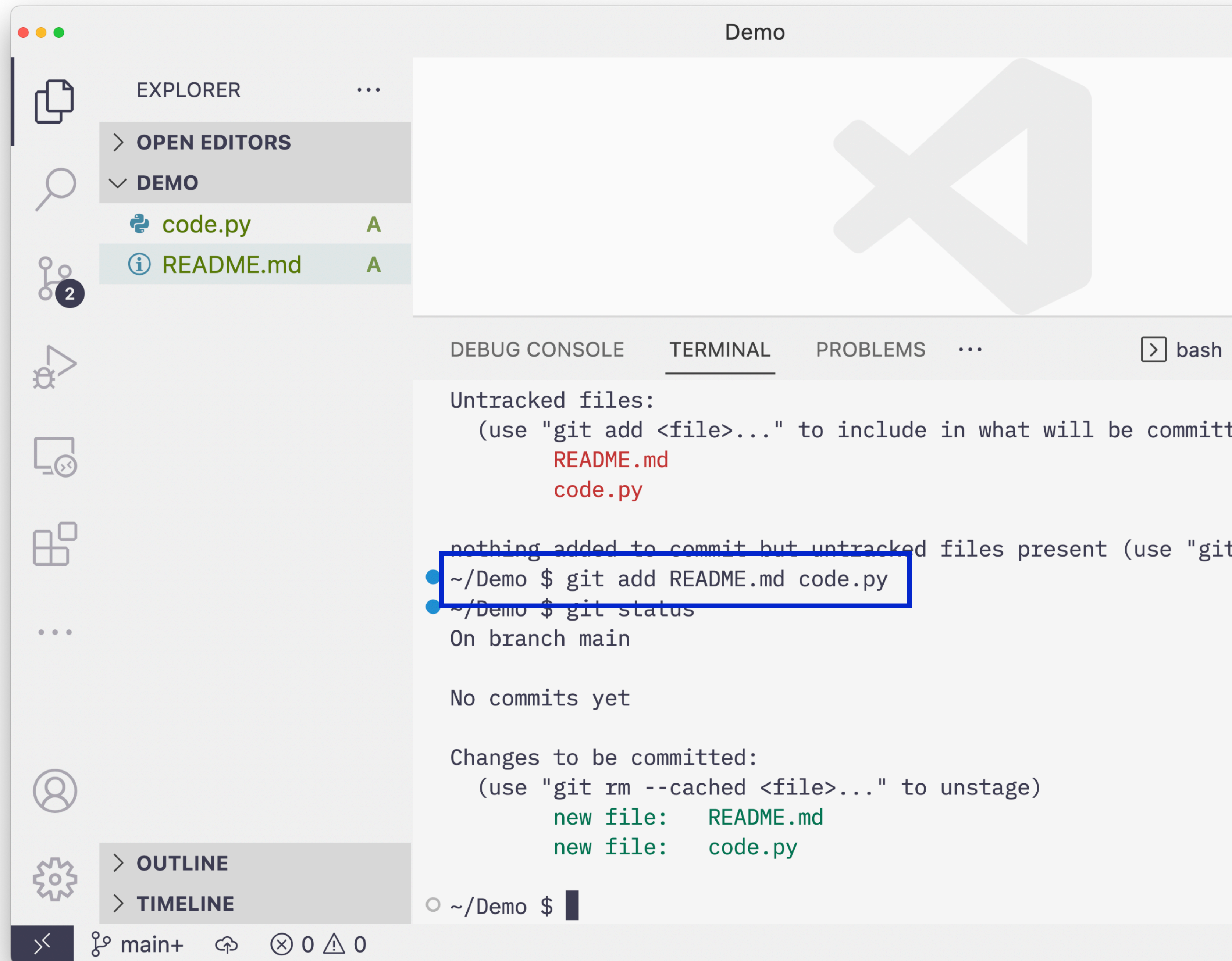
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        README.md
        code.py

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

git basics

Setup

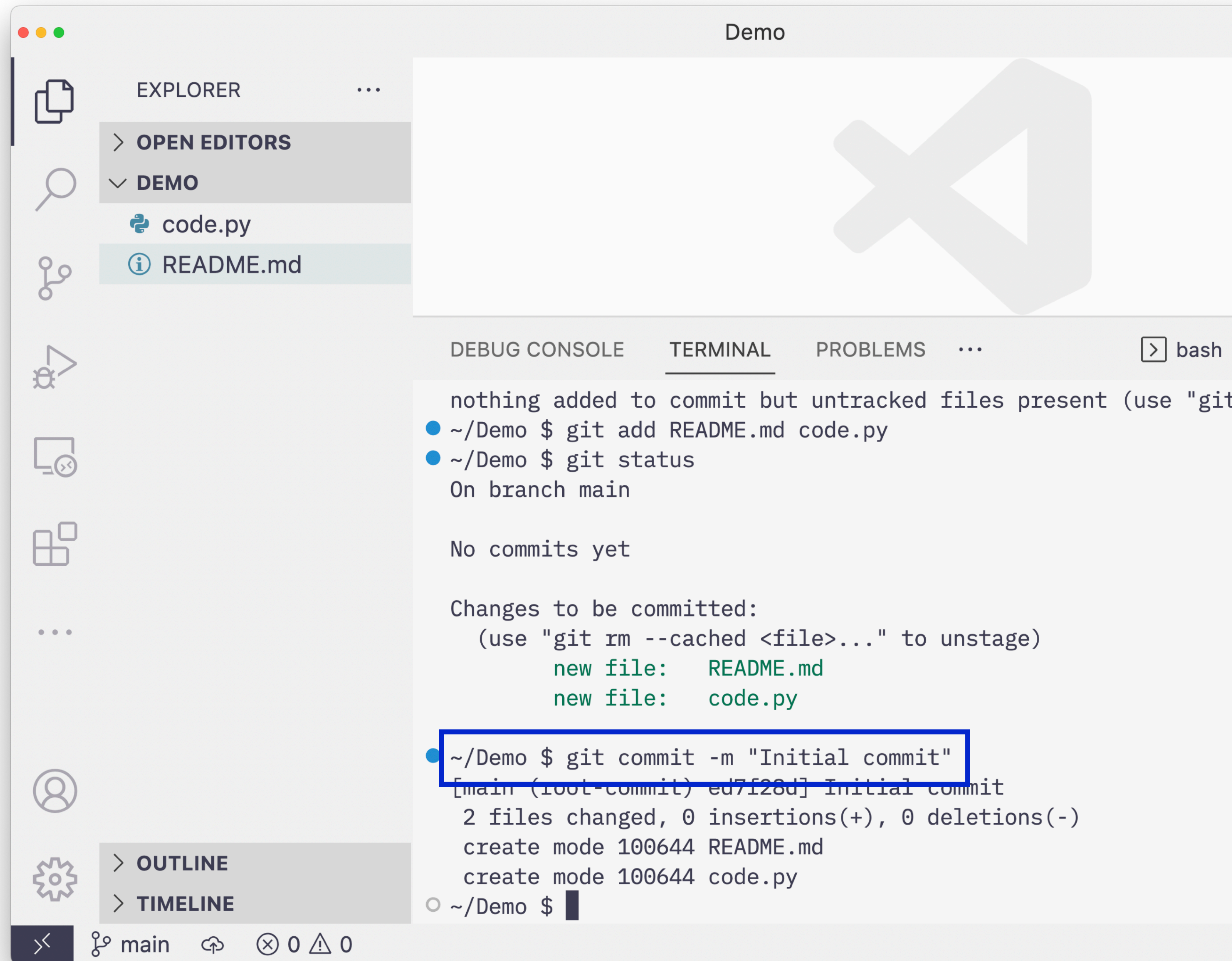
- Use `git add` to stage new or changed files



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`



The screenshot shows the Visual Studio Code interface with a terminal window open. The Explorer sidebar on the left shows the file structure for a project named 'DEMO', including 'code.py' and 'README.md'. The terminal window displays the following output:

```
Demo

DEBUG CONSOLE  TERMINAL  PROBLEMS  ...  bash
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

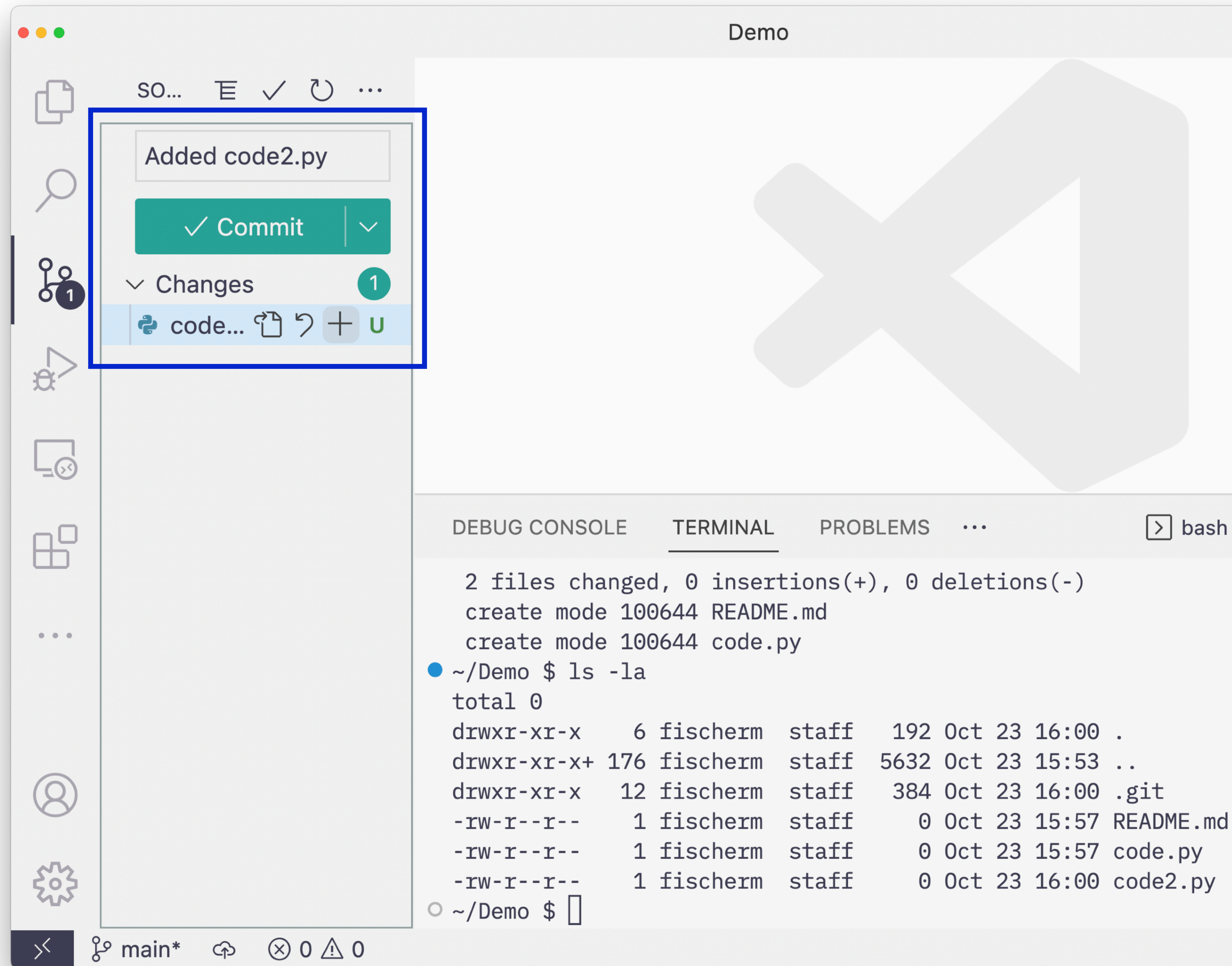
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
    new file:   README.md
    new file:   code.py

~/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

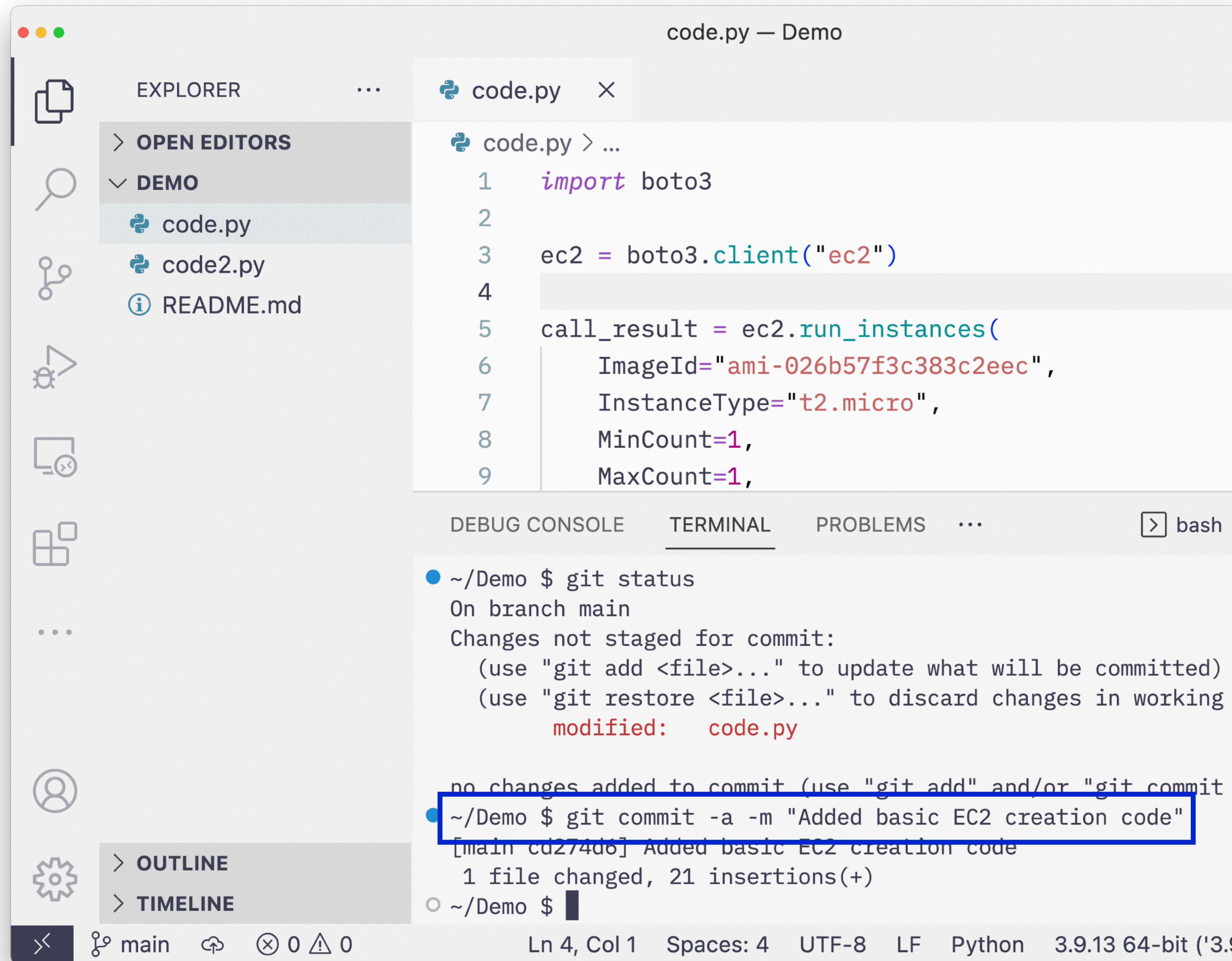
- Tools like VS Code have built-in support for `git`
- Add and commit changed files directly in VS Code GUI



git basics

Setup

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



The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left shows a project named 'DEMO' with files 'code.py', 'code2.py', and 'README.md'. The main editor window displays the contents of 'code.py', which contains Python code for creating an EC2 instance using boto3. The terminal window at the bottom shows the output of the following commands:

```
~/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

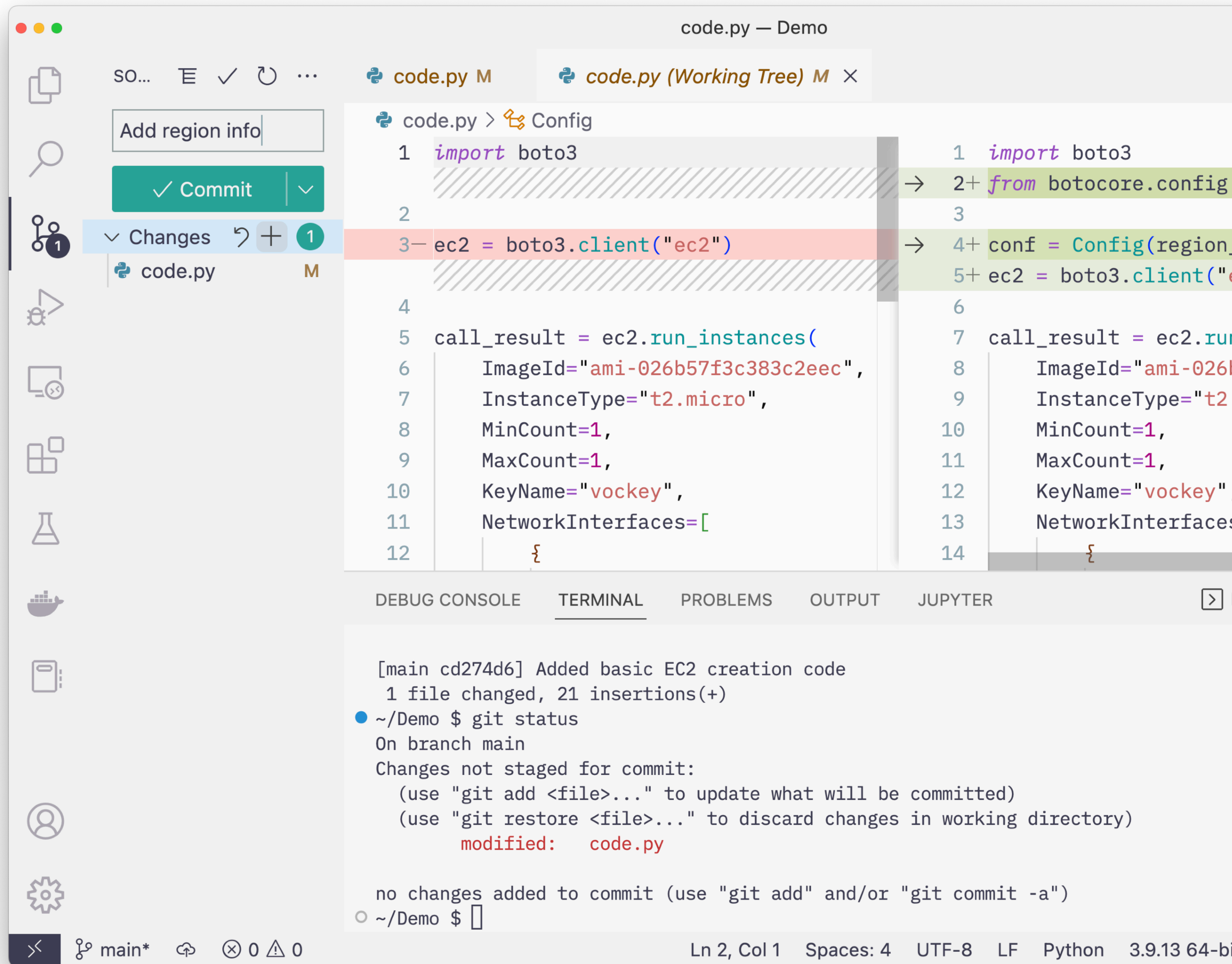
no changes added to commit (use "git add" and/or "git commit" to update what will be committed)
~/Demo $ git commit -a -m "Added basic EC2 creation code"
[main cd274d6] Added basic EC2 creation code
1 file changed, 21 insertions(+)
```

The terminal output is partially obscured by a blue box highlighting the command and its output. The status bar at the bottom indicates the current branch is 'main', there are 0 errors and 0 warnings, and the file is at line 4, column 1.

git basics

Setup

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



The screenshot displays the VS Code interface with a diff view of a Python file named `code.py`. The left pane shows the original code, and the right pane shows the modified code. The diff highlights the changes: line 3 is removed (red background), and lines 2-5 are added (green background). The terminal at the bottom shows the output of the `git status` command, indicating that the file `code.py` has been modified.

```
code.py — Demo
code.py M code.py (Working Tree) M X
code.py > Config
1 import boto3
2
3- ec2 = boto3.client("ec2")
4
5 call_result = ec2.run_instances(
6     ImageId="ami-026b57f3c383c2eec",
7     InstanceType="t2.micro",
8     MinCount=1,
9     MaxCount=1,
10    KeyName="vockey",
11    NetworkInterfaces=[
12        {
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

```
DEBUG CONSOLE TERMINAL PROBLEMS OUTPUT JUPYTER
[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 $
```

Ln 2, Col 1 Spaces: 4 UTF-8 LF Python 3.9.13 64-bit

git basics

Setup

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

The screenshot shows the Visual Studio Code interface with a Python file named `code.py` open in the editor. The Explorer pane on the left shows the file structure with `code.py`, `code2.py`, and `README.md`. The Timeline pane at the bottom left shows the commit history for `code.py`, with three commits listed: "Add region info" (2 mins), "Added basic EC2 creatio..." (4 mins), and "Initial commit" (8 mins). The Terminal pane at the bottom right shows the output of the `git log` command, displaying the commit history in a structured format.

```
code.py — Demo
code.py > ...
1 import boto3
2 from botocore.config import Config
3
4 conf = Config(region_name="us-east-1")
5 ec2 = boto3.client("ec2", config=conf)
6
7 call_result = ec2.run_instances(
8     ImageId="ami-026b57f3c383c2eec",
9     InstanceType="t2.micro",
)

~/Demo $ git log
commit 40c91b48b00f9561d0c1e3ef2b1cf7e56de7f73 (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 $
```