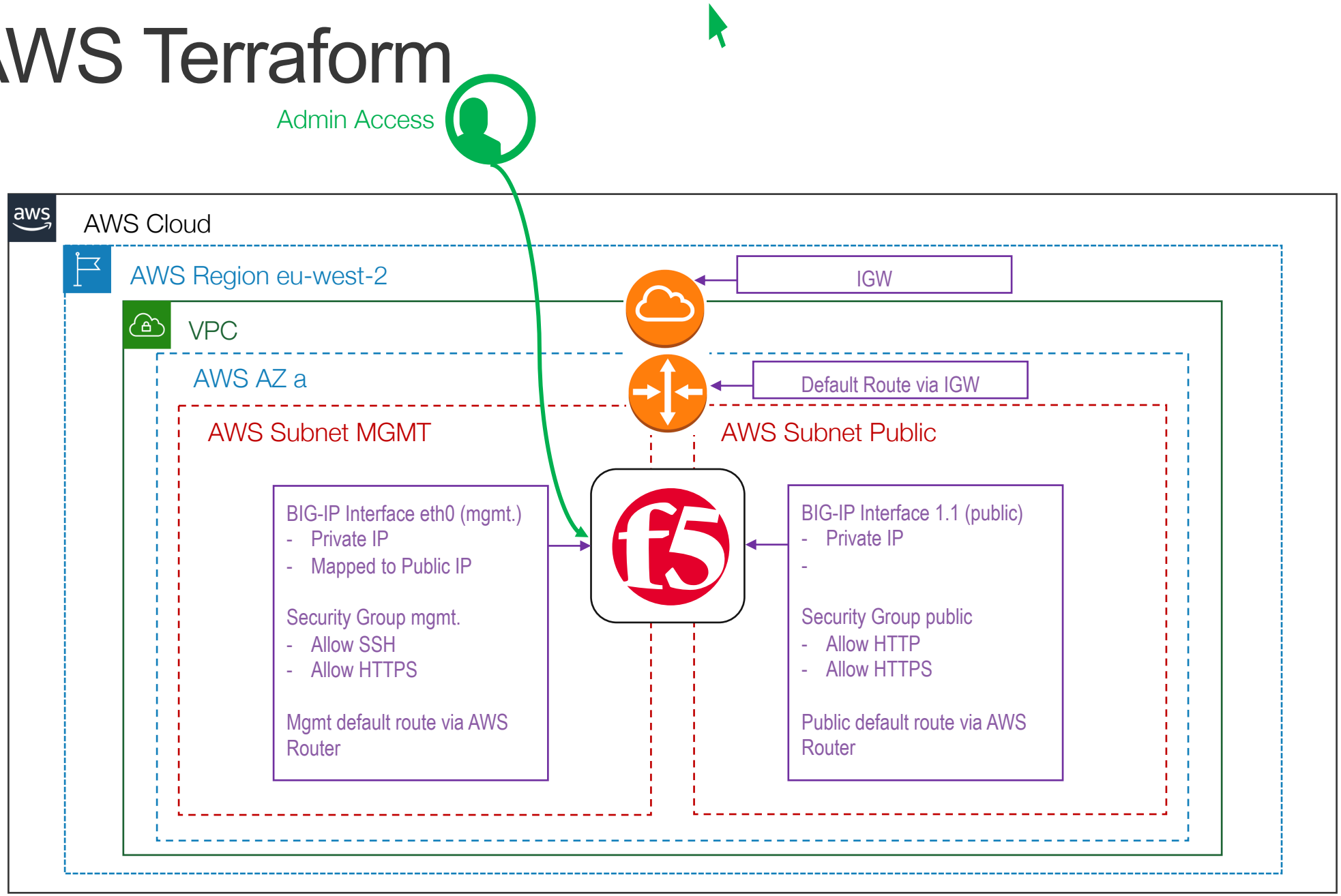
Automation Workshop (201)

How to automate like a pro!

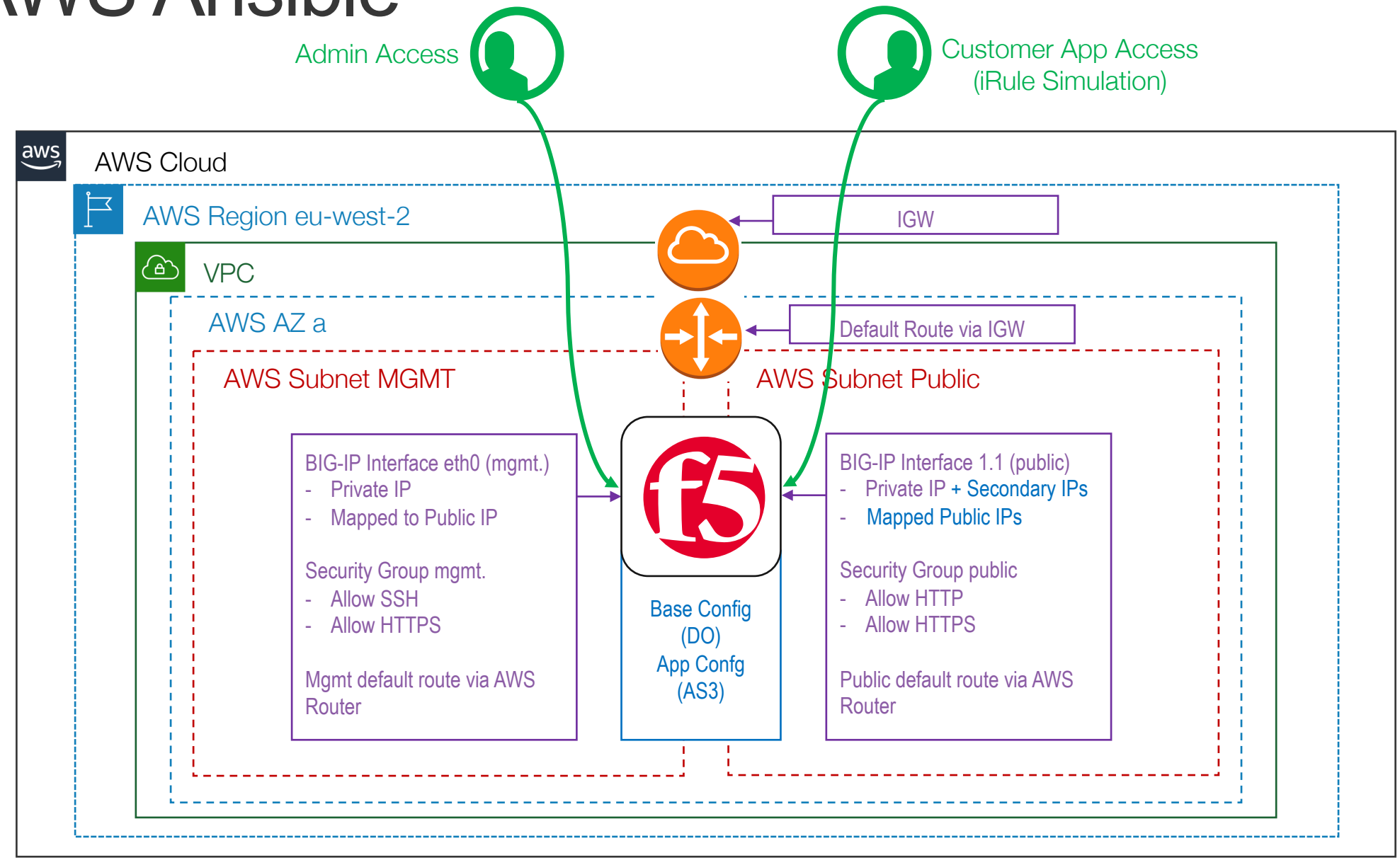
Prerequisites

- **Some more information on the software required for the workshop**
- All Windows and Mac Users:
  - Microsoft VS Code: https://code.visualstudio.com/download
  - Git client: https://git-scm.com/download
  - Register for Slack, https://slack.com/intl/en-gb/ and optionally, install the client for your OS.
  - Register and create a github account: https://github.com/
  - Register and create an AWS account: https://aws.amazon.com/free
    - …or use your existing AWS account, if you have appropriate permissions to create IAMs, VPCs & deploy EC2 instances.
  - Install the following libraries (using pip, apt-get, etc): python3-jmespath, boto, boto3, botocore, netaddr
- Mac users:
  - Ansible: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html
  - AWS CLI https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html
  - Terraform: https://www.terraform.io/downloads.html
- Windows 10 Users:
  - Windows Subsystem Linux (Ubuntu): https://docs.microsoft.com/en-us/windows/wsl/install-win10
  - Install Ansible into WSLinux: https://www.jeffgeerling.com/blog/2017/using-ansible-through-windows-10s-subsystem-linux
  - AWS CLI https://docs.aws.amazon.com/cli/latest/userguide/cli-chap-install.html - Make sure you have installed AWS CLI into WSLinux rather than PowerShell.
  - Terraform: https://www.terraform.io/downloads.html - Make sure you have installed Terraform into WSLinux rather than PowerShell.
- Windows 7 Users:
  - Cygwin & Ansible: https://everythingshouldbevirtual.com/automation/ansible-using-ansible-on-windows-via-cygwin/
  - Or… use a Linux VM and install Ansible.

F5 AWS Terraform

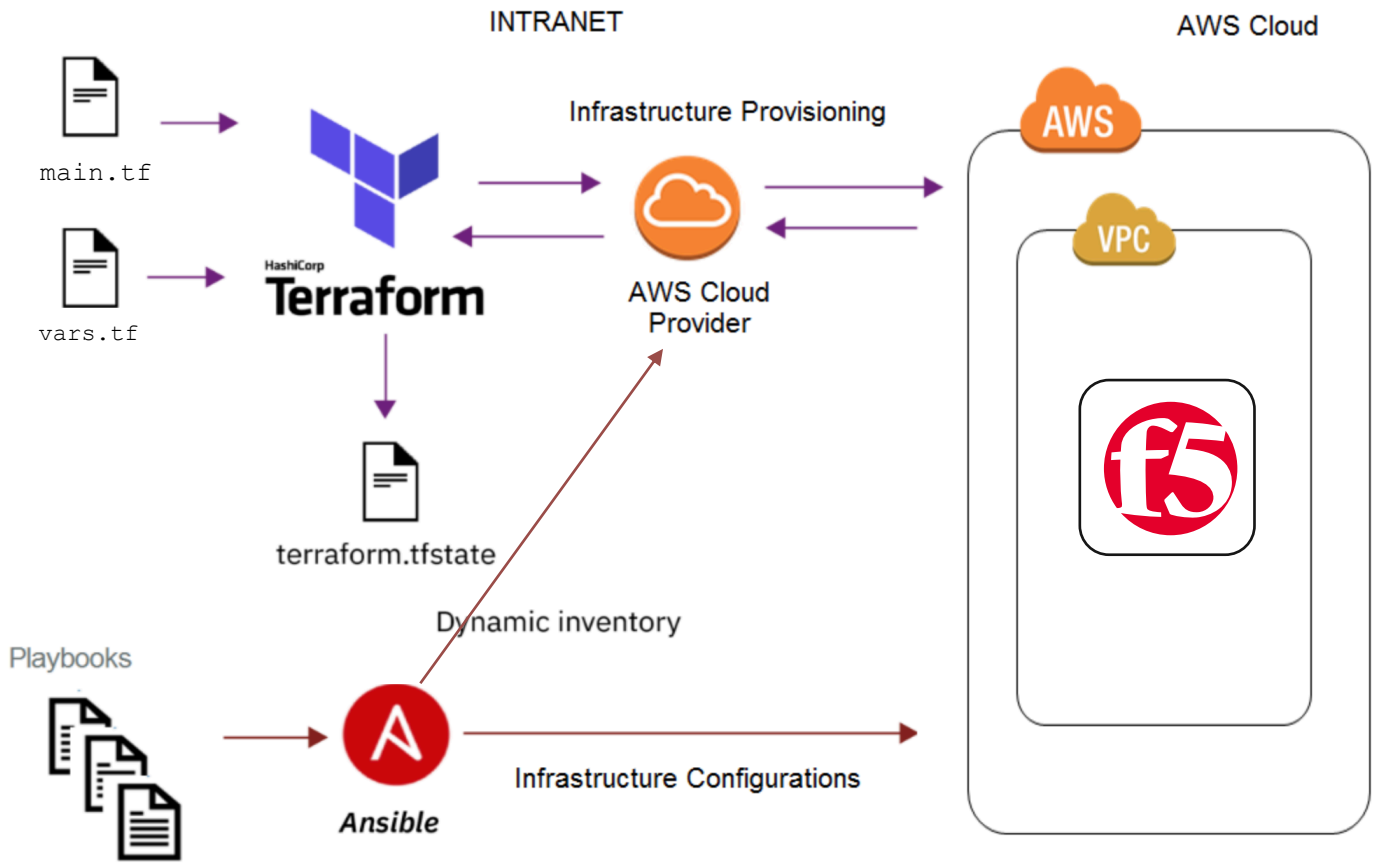


F5 AWS Ansible



Terraform & Ansible

- Terraform = Infrastructure as Code



- Ansible = Configuration Management

Terraform

- Infrastructure scaffolding (logical “environment”, servers, network, storage, etc.)
- Inbuilt state management (tracks the state of your infrastructure and the impact)
- Declarative style execution (code that specifies your desired end state)
- Does not rollback on failure (if some part of your deployment fails unfortunately it does not know how to roll back the changes).
- Maintains the version history of the infrastructure
- Use a client-only architecture

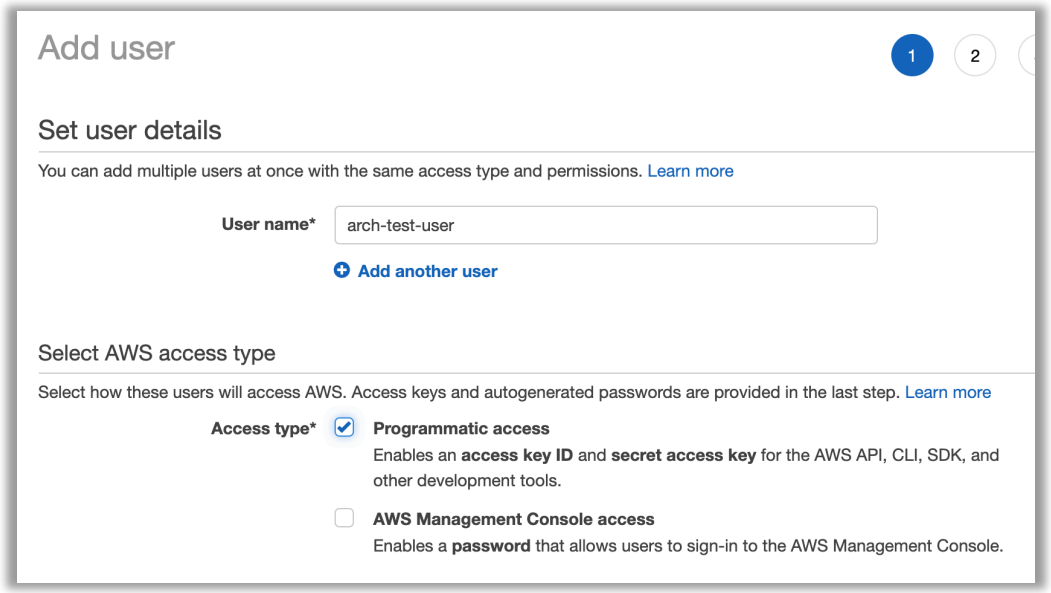
Ansible

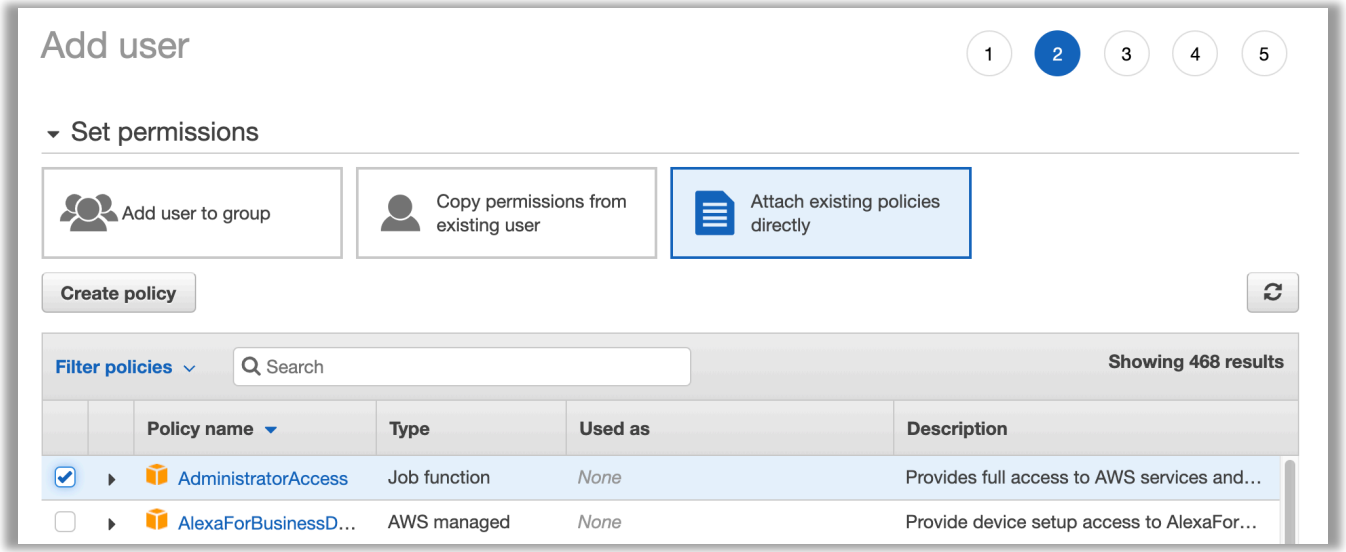
- Procedural describes an application that requires the exact steps to be laid out in the code.
- It’s procedural language (step-by-step) application using YMAL
- Deploy once and you are done, unless you run it again and the outcome is Idempotent
- It’s agent-less and uses SSH communication
- Use a client-only architecture

Register with AWS

Create in eu-west-2 (London)

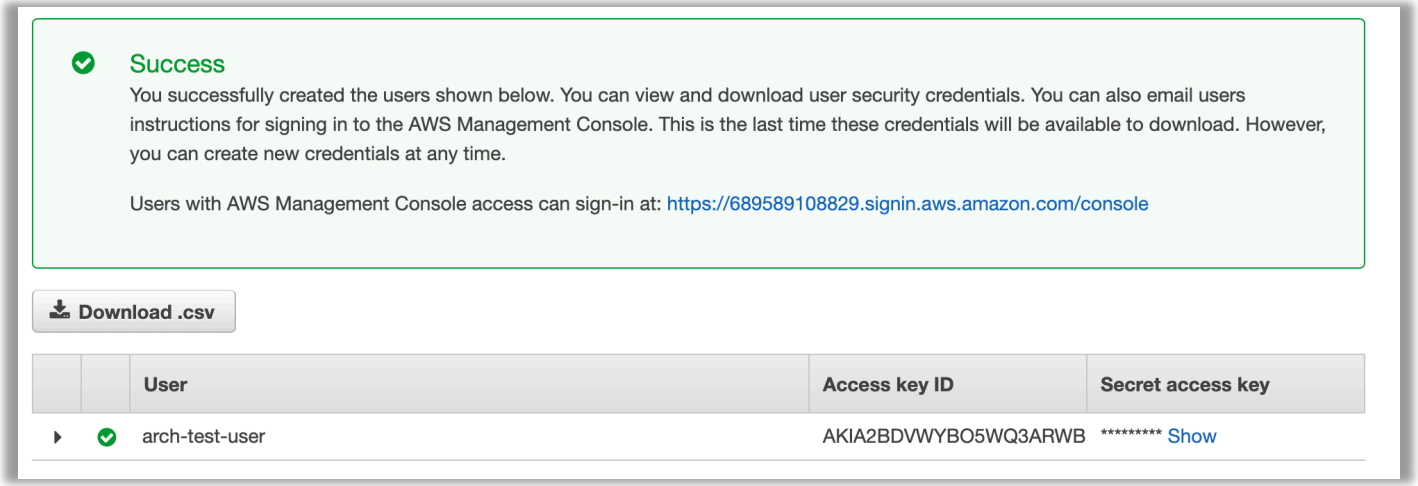
- https://aws.amazon.com/free
- Create a new account… this might link to your Amazon account. Add payment info if required.
- Create IAM, New User, for programmatic access, make admin, next, next, create:
- MAKE SURE YOU ARE IN REGION eu-west-2





Save AWS Creds

- Copy the Access key ID and secret access key.
- Create ~/creds/aws_creds.yaml



- Declare the two values as Ansible variables ‘ec2_access_key’ & ‘ec2_secret_key’ (ideally vault encrypted – see next slide):

```
# AWS API Credentials.
# Use following command if you want to encrypt (strongly recommended).
# ansible-vault encrypt_string 'M6HT0QWsFAjQ*********' --name 'ec2_secret_key'
# run the ansible playbook with --ask-vault-pass flag.
# Else, use unencrypted values like this:

    ec2_access_key: AKIA2BDVWYBO5WQ3ARWB
    ec2_secret_key: M6HT0QWsFAjQ********
```

Encrypt Creds - Ansible Vault

- Create credentials file aws_creds.yaml

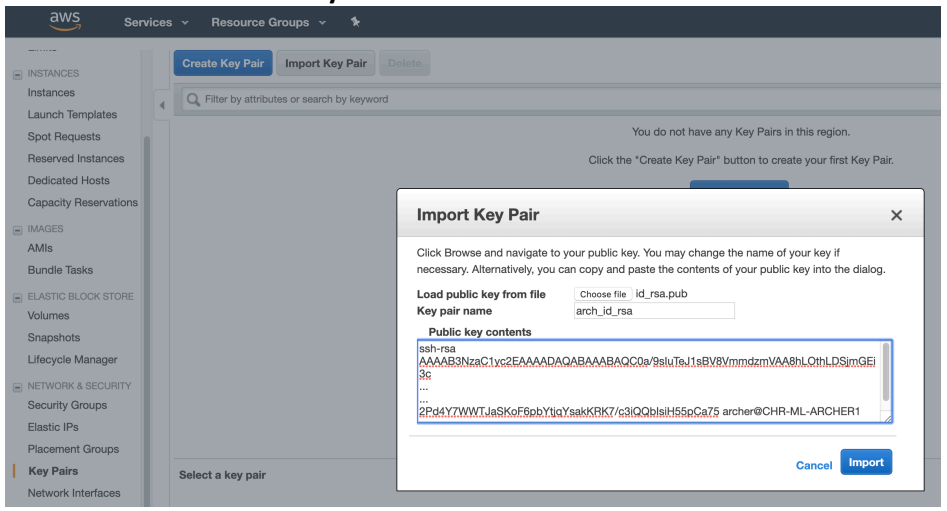
```
# AWS API Credentials
# ansible-vault encrypt_string 'my_password!' --name 'my_pass_var_name'
# run the ansible playbook with --ask-vault-pass flag.

ec2_access_key: !vault |
$ANSIBLE_VAULT;1.1;AES256
30363234383334343738343130633330376431653031343433646231663661666239376230336233
62613961353535653564636264326532613939306561653 40a333365366438623064653562336339
61363739373762376131356233376161333631363463653439613331303238616331313261666664
33303238303337306 20a663062643633383239383662386138623364663539316335626134363663
3165623566633363333763383437646232643331323762643332343566323833346 1
ec2_secret_key: !vault |
$ANSIBLE_VAULT;1.1;AES256
62346262353566623162313061626232643165633532376636313633373232353731326366343661
39643662313465636465313739353236613532333834343 00a323264373162643361396466663936
65633466653332656638376562373030316565326364383938363730346263373534643336363233
36383636623033613 80a326366636630333562666266653838343531353266336637656433333030
62353039393132396637306563396265643133643163353232396563303861333162376439613636
34366363313332306134323963616164363364636666393537 63
```

```
.
├── autows201
│   ├── README.md
│   ├── ansible
│   │   ├── inventory
│   │   ├── playbooks
│   │   │   ├── destroy.yaml
│   │   │   └── main.yaml
│   │   ├── tasks
│   │   │   ├── tasks-01-aws-eni-ips.yaml
│   │   │   ├── tasks-02-deploy-do.yaml
│   │   │   ├── tasks-03-deploy-as3.yaml
│   │   │   ├── tasks-04-outputs.yaml
│   │   │   └── tasks-11-remove-aws-eni-ip
│   │   ├── templates
│   │   │   ├── as3-declaration.j2
│   │   │   └── do_base.j2
│   │   └── vars
│   │       └── vars.yaml
│   └── terraform
│       ├── main.tf
│       ├── terraform.tfstate
│       ├── terraform.tfstate.backup
│       └── vars.tf
└── creds
    ├── aws_creds.yaml
    └── big_creds.yaml
```

Import SSH Key

Import in eu-west-2 (London)

- You need to IMPORT your public SSH key **for every region** you want to deploy EC2 instances.
- Select the file from your local machine and give it a meaningful name for use in AWS (you don't need to use the same name as the filename on your local machine.



- If you don't have an SSH private key… create an SSH key pair in PEM format (or use your existing key pair e.g. ~/.ssh/id_rsa[.pub]) -- ssh-keygen -t rsa # Accept the default names and do not set a passphrase. If you're running Python 2.7, you might need to run ssh-keygen -t rsa -m PEM and append .pem to the filename.

Find AMI Using AWS CLI

- Make sure you have installed the AWS CLI Tool with credentials & config in the ~/.aws folder (https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-files.html)
  - If creds are not there, run 'aws configure'
- Run the following command to find the F5 AMI (Amazon Machine Image) for F5 BIGIP BEST 25 Mbps in region eu-west-2 (London). Maybe

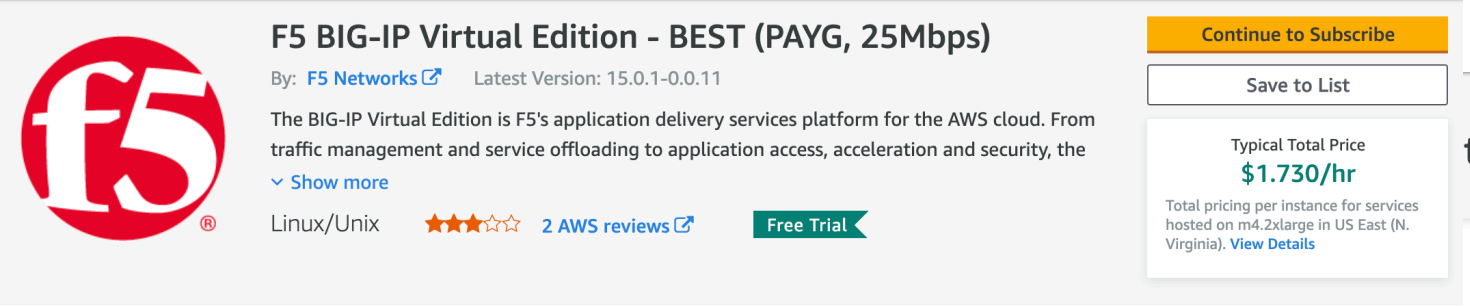
```
aws ec2 describe-images --region eu-west-2 --filters "Name=name,Values=*BIGIP-14.1.2*PAYG-Best*25*" |
grep '\"Name\"\|\"ImageId\”’
```

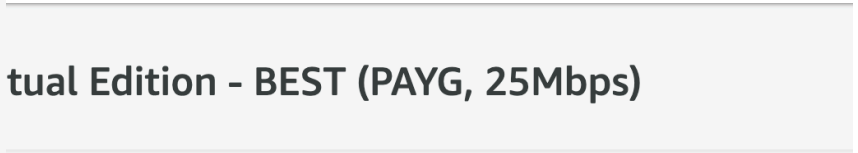
```
aws2 ec2 describe-images --region eu-west-2 --filters "Name=name,Values=[*BIGIP-14.1.2*PAYG-Best*25*]"
| grep '\"Name\"\|\"ImageId\"'
```

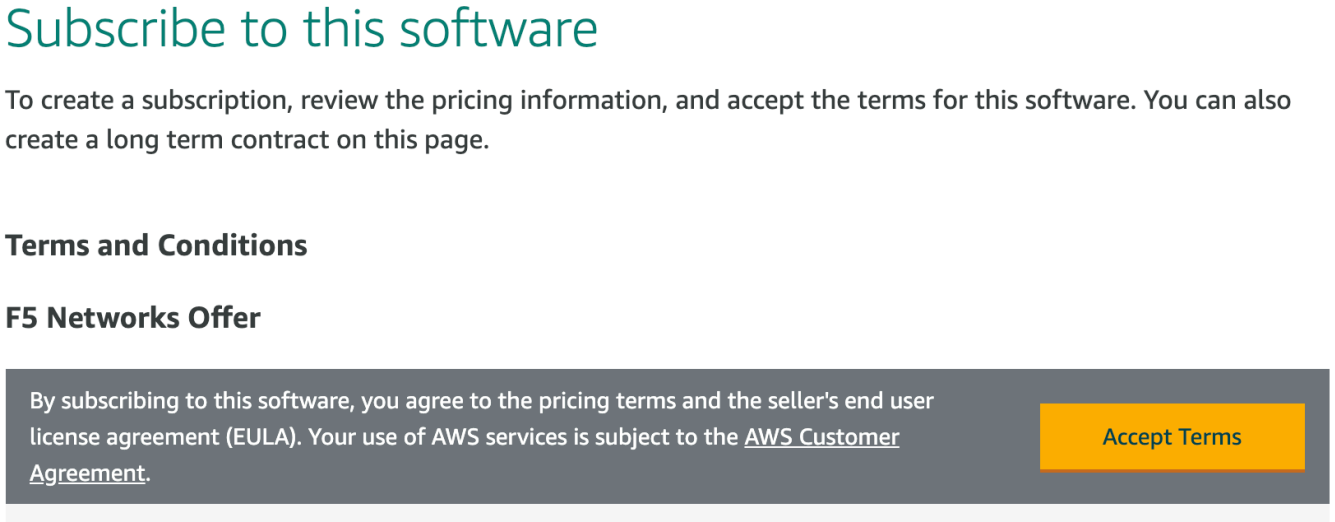
```
$ aws ec2 describe-images --region eu-west-2 --filters "Name=name,Values=*BIGIP-14.1.2*PAYG-Best*25*" | grep
'\"Name\"\|\"ImageId\"'



"ImageId": "ami-04ae6503b1b20981c",

"Name": "F5 BIGIP-14.1.2-0.0.37 PAYG-Best 25Mbps-190807130231-3e567b08-20a9-444f-a72a-7e8da3c2cbdf-ami-
01a28e890dc5b04fe.4"
```

Subscribe to F5 in AWS Market Place

- Go to the Market Place https://console.aws.amazon.com/marketplace/home?region=us-east-1#/search
- Search for ‘F5 BIGIP PAYG-Best 25Mbps’ and subscribe to the software.







Create AWS Secret

Create in eu-west-2 (London)

AWS Secrets Manager > Secrets > Store a new secret

Store a new secret

Select secret type Info

- Credentials for RDS database
- Credentials for Redshift cluster
- Credentials for DocumentDB database
- Credentials for other database
- Other type of secrets (e.g. API key)

Specify the key/value pairs to be stored in this secret Info

Secret key/value | Plaintext

Pa$$word123

AWS Secrets Manager > Secrets > Store a new secret

Store a new secret

Secret name and description Info

Secret name

Give the secret a name that enables you to find and manage it easily.

my_bigip_password

Secret name must contain only alphanumeric characters and the characters /_+=.@-

Description - *optional*

e.g - Access to MYSQL prod database for my AppBeta

Maximum 250 characters

Tags - *optional*

Key | Value - *optional*

Enter key | *Enter value* | Remove

Add

Cancel | Previous | Next

Create Git Repo (autows201)

- Visit https://github.com and create repo autows201
- …and clone to client machine. e.g. git clone https://github.com/s-archer/autows201.git
- Open VS code and open a terminal in the new git folder…
- Edit .gitignore and add:

  *.tfstate
  *.tfstate.backup
  *.retry
  creds
  password

- Create folder structure:

```
.
├── ansible
│   ├── inventory
│   ├── playbooks
│   ├── tasks
│   ├── templates
│   └── vars
└── terraform
```

Create Simple Terraform Config

- https://Terraform.io getting started guide.
  - On ‘Build Infra’ page, note that AWS credentials used by TF are same as used by AWS CLI.
  - Copy the example code into a new file called main.tf in your git folder UPDATE THE REGION!

```
provider "aws" {
  profile = "default"
  region = "us-east-1"
}
resource "aws_instance" "example" {
  ami = "ami-2757f631"
  instance_type = "t2.micro"
}
```

- Run ‘terraform init’ to get required providers.
- Run ’terraform apply’ to deploy.
- Run ‘terraform destroy’ to remove.

Add VPC to Terraform Config

- https://www.terraform.io/docs/providers/aws/r/vpc.html – or navigate from terraform.io to docs >> providers >> AWS >> VPC (list on left) >> Resources >> aws_vpc
- Add config from the example below (and entirely replace the previous examples code):

```
resource "aws_vpc" "main" {
  cidr_block       = "10.0.0.0/16"
  instance_tenancy = "dedicated"

  tags = {
    Name = "main"
  }
}
```

- Run ‘terraform init’ to get required providers.
- Run ’terraform apply’ to deploy.

Create in AZ eu-west-2a

Add Subnets to Terraform Config

- https://www.terraform.io/docs/providers/aws/r/subnet.html – or navigate from terraform.io to docs >> providers >> AWS >> VPC (list on left) >> Resources >> aws_subnet
- Add config from the example, and duplicate to create two subnets: 'mgmt' and 'data':
- !Need to add 'availability_zone = "eu-west-2a"' to the example below!

Basic Usage

```
resource "aws_subnet" "main" {
  vpc_id     = "${aws_vpc.main.id}"
  cidr_block = "10.0.1.0/24"

  tags = {
    Name = "Main"
  }
}
```

22 & 443 for MGMT, 80 & 443 for Public

Add Security Groups to Terraform Config

- https://www.terraform.io/docs/providers/aws/r/security_group.html – or navigate from terraform.io to docs >> providers >> AWS >> VPC (list on left) >> Resources >> aws_security_group
- Add config from the example, and duplicate to create two security groups: 'mgmt' and 'data':

```
resource "aws_security_group" "allow_tls" {
  name        = "allow_tls"
  description = "Allow TLS inbound traffic"
  vpc_id      = "${aws_vpc.main.id}"

  ingress {
    # TLS (change to whatever ports you need)
    from_port   = 443
    to_port     = 443
    protocol    = "tcp"
    # Please restrict your ingress to only necessary IPs and ports.
    # Opening to 0.0.0.0/0 can lead to security vulnerabilities.
    cidr_blocks = # add a CIDR block here
  }

  egress {
    from_port       = 0
    to_port         = 0
    protocol        = "-1"
    cidr_blocks     = ["0.0.0.0/0"]
    prefix_list_ids = ["pl-12c4e678"]
  }
}
```

Add IGW & Routes to Terraform Config

- Create two subnets (mgmt. & public). (!Need to add ‘availability_zone = "eu-west-2a”’!)
- Create two security groups (mgmt. & public)
- Add Internet Gateway https://www.terraform.io/docs/providers/aws/r/ internet_gateway.html
- Add Route Table https://www.terraform.io/docs/providers/aws/r/route_table.html
- Associate Route Table with Subnets https://www.terraform.io/docs/providers/aws/r/route_table_association.html
- See next slide for summary table:

Add Resources Terraform Config

	Resource	Name (TF)	Name (AWS)	Values
1	aws_vpc	main	arch-vpc	10.0.0.0/16
2	aws_subnet	mgmt	mgmt	10.0.1.0/24
3	aws_subnet	public	public	10.0.2.0/24
4	aws_security_group	mgmt	mgmt	Ingress 22, 443, 0.0.0.0/0; Egress 0, 0, 0.0.0.0/0
5	aws_security_group	public	public	Ingress 80, 443, 0.0.0.0/0; Egress 0, 0, 0.0.0.0/0
6	aws_internet_gateway	gw	main	n/a
7	aws_route_table	main-rt	main	0.0.0.0/0 via IGW ID
8	aws_route_table_association	mgmt	n/a	Associate with ${ aws_subnet.mgmt.id }
9	aws_route_table_association	public	n/a	Associate with ${ aws_subnet.public.id }

Add BIGIP AMI Deploy to Terraform Config

- https://github.com/f5devcentral/terraform-aws-bigip#example-2-nic-deployment-payg
- Add example code from ‘example 2-NIC Deployment PAYG

Example 2-NIC Deployment PAYG

```
module bigip {
  source = "f5devcentral/bigip/aws"
  version = "0.1.2"

  prefix            = "bigip"
  f5_instance_count = 1
  ec2_key_name      = "my-key"
  aws_secretmanager_secret_id = "my_bigip_password"
  mgmt_subnet_security_group_ids = [sg-01234567890abcdef]
  public_subnet_security_group_ids = [sg-01234567890ghijkl]
  vpc_mgmt_subnet_ids = [subnet-01234567890abcdef]
  vpc_public_subnet_ids = [subnet-01234567890ghijkl]
  # NEED TO ADD BELOW TO REPLACE DEFAULT IN MODULE
  f5_ami_search_name = "F5 Networks BIGIP-14.* PAYG - Best 25*”
}
```

Add Outputs to Terraform Config

- Add outputs (look at the outputs in the bigip module config (need to ‘terraform init’ config first)
- Outputs defined in module can be accessed using module.<module name>.<output name> for example:

- bigip Module outputs this (silently):

```
output "mgmt_public_ips" {
  description = "List of BIG-IP public IP addresses for the management interfaces"
  value = aws_eip.mgmt[*].public_ip
}
```

- We can access the output, and put to stdout with this:

```
output "mgmt_public_ips" {
  description = "List of BIG-IP public IP addresses for the management interfaces"
  value       = module.bigip.mgmt_public_ips
}
```

Begin Ansible Playbook

```
.
├── autows201
│   ├── README.md
│   ├── ansible
│   │   ├── inventory
│   │   ├── playbooks
│   │   │   ├── destroy.yaml
│   │   │   └── main.yaml
│   │   ├── tasks
│   │   │   ├── tasks-01-aws-eni-ips.yaml
│   │   │   ├── tasks-02-deploy-do.yaml
│   │   │   ├── tasks-03-deploy-as3.yaml
│   │   │   ├── tasks-04-outputs.yaml
│   │   │   └── tasks-11-remove-aws-eni-ips.yaml
│   │   ├── templates
│   │   │   ├── as3-declaration.j2
│   │   │   └── do_base.j2
│   │   └── vars
│   │       └── vars.yaml
│   └── terraform
│       ├── main.tf
│       ├── terraform.tfstate
│       ├── terraform.tfstate.backup
│       └── vars.tf
└── creds
    ├── aws_creds.yaml
    └── big_creds.yaml
```

Add Ansible Inventory Creation to Terraform Config

```
resource "local_file" "bigips_inventory" {
  content = <<EOF
[bigips]
${module.bigip.mgmt_public_ips.0}
aws_pub_eni_id=${module.bigip.public_nic_ids.0}
EOF
  filename = "../ansible/inventory/bigips.ini"
}
```