Basic Networking for Cloud Platforms: Essential Network Functions, Security, Access Control Part II

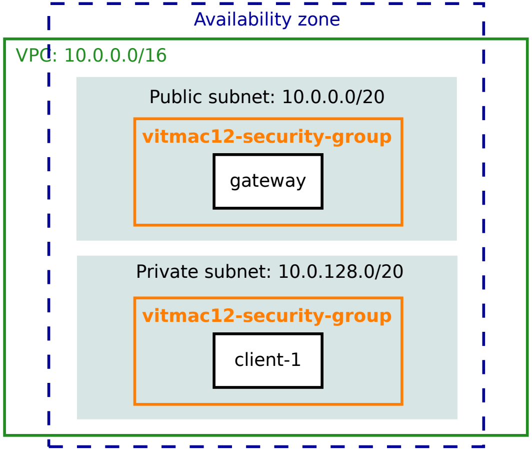
Balázs Sonkoly, Felicián Németh, István Pelle, Balázs Fodor

2024-10-03

Infrastructure as code: a very high-level overview

- Infrastructure as code (IaC):
  - Managing and provisioning cloud resources using definition files
  - Instead of interactive configuration tools
- Advantages:
  - Faster than manual configuration (you'll see), can be put under version control (it's text, so obviously)
- Disadvantages:
  - Learning curve, time to create definition files can be much longer than setting up the infrastructure manually
- Some notable examples:
  - AWS-specific: CloudFormation (coming up next), Cloud Development Kit (higher-level)
  - Provider-agnostic: Terraform/OpenTofu, Pulumi
    - For serverless (FaaS) compute: the Serverless Framework

Proprietary task



Automated infrastructure setup (1/7): AWS CloudFormation template

Idea

- We use AWS CloudFormation for handling most of the infrastructure setup for us
- CloudFormation requires a JSON or YAML template: we provide JSON but it can be converted to YAML
- The AWS Cloud Development Kit (CDK) would be a better, higher-level option of setting up the infrastructure, but it is not available in the Learner Lab environment
- Platform-agnostic tools are also available – we won't work with any of those for now

Setup steps

- Download the AWS CloudFormation from the following link: `https://tinyurl.com/vitmac12-week04-template`
  - There's an alternative method for the case when file download is not allowed
  - If you downloaded the file, look inside it: search for the `Resources` element and observe that many elements are there that we have previously configured manually (observe also that there are some unexpected elements, e.g., those with the name of `association`)

Automated infrastructure setup (2/7): AWS CloudFormation template for IMSc students

Additional steps for IMSc students:

- Open the template and study it
    - No alternative for this, unfortunately
- Search for `Client1` among `Resources`
- Compare the resource definition to the reference at https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-resource-ec2-instance.html
- Modify the template to include a second client:
    - Use `Client2` as the resource name
    - Set `client-2` as the client's name (under `Properties` → `Tags` see `"Key": "Name"` and the corresponding `"Value"`)
    - Set `10.0.128.20` as `client-2`'s IP address (see the `PrivateIpAddress` property)

Infrastructure setup (3/7): Deployment using AWS CloudFormation (1/2)

Setup steps

- In AWS Management Console navigate to the CloudFormation service:
  - Type cf in the search bar and select CloudFormation
- In the CloudFormation console click on the Create stack button
  - If a drop down list appears under the button, select the With new resources (standard) option
  - In `Step 1 Create stack`, under the `Prerequisite - Prepare template` section select the `Choose an existing template` option
  - In the `Specify template` section select the `Upload a template file` option, click the `Choose file` button and select the previously downloaded template file
    - Alternatively, select the `Amazon S3 URL` option and copy the following link to the `Amazon S3 URL` text box: `https://vitmac12-resources.s3.amazonaws.com/practice-04.template`
  - Click the `Next` button
  - In `Step 2 Specify stack details`, under the `Provide a stack name` section set the following name: `vitmac12`
  - Leave the other parameters unmodified if you don't have a previous active deployment of the same practice session infrastructure either from a manual setup of from a CloudFormation deployment
    - If you have an active deployment, under `Parameters` → `Prefix` modify the provided different prefix (maybe by appending a letter at the end)
  - Click the `Next` button

Infrastructure setup (4/7): Deployment using AWS CloudFormation (1/2)

Setup steps continued:

- In `Step 3 Configure stack`, under the `Permissions` section keep the `IAM role name` in the first drop down list, in the second drop down list select `LabRole`
- Leave the other options unchanged
- Scroll down and click the `Next` button
- In `Step 4 Review and create`, scroll down and click the `Submit` button
- Wait until the stack reaches the `CREATE_COMPLETE` state
  - If something goes wrong, the stack rolls back (but this is not expected to happen)
- While keeping the `vitmac12` stack selected, click the `Outputs` tab
  - Make a note on the name of the VPC and the security group that you've just created (you will use them later)

Infrastructure setup (5/7): Manual configuration – private network interface

Idea

- CloudFormation did most of the job but some parts are still missing
  - Due to time and complexity issues
- Some configurations cannot be done in a single template
  - Updates (changes sets) can be applied: this is out of the scope for this practice session
- We manually assign the interface in the private subnet to the gateway

Setup steps

- Go to EC2 → Instances:
  - Click the EC2 refresh button if you don't see the newly created gateway and client instances
  - Wait until the newly created instances reach the `Running` state with `2/2 checks passed` status check (you might need to use the EC2 refresh button to see the change)
  - Select the `gateway` instance (checkmark it)
  - Click the `Actions` button (top right) then select `Networking` → `Attach network interface`
    - Under `VPC`, select the VPC that CloudFormation has created for you (refer to the CloudFormation stack's outputs)
    - Under `Network interface`, select `gateway-private-int` that CloudFormation has created for you
    - Click the `Attach` button

Infrastructure setup (6/7): Manual configuration – security group

Idea

- CloudFormation did most of the job but some parts are still missing
- We also enable the traffic inside the security group manually

Setup steps

- At EC2 → Instances, keep the `gateway` instance selected and select the `Security` tab down below
  - Under `Security groups` click the name of the security group (the name and ID should match those shown among the CloudFormation stack's outputs)
  - In the `Inbound rules` tab click the `Edit inbound rules` button
    - Keep the rule that is already there and click the `Add rule` button
    - For `Type` set `All traffic`, for `Source` keep `Custom`, and click the text field with the magnifying glass beside it and select the security group that CloudFormation created for you (you can refer to the CloudFormation stack's outputs), set the description to `Allow all traffic within the security group`
    - Click the `Save rules` button

Infrastructure setup (7/7): Connect to your instances

- Go back to your gateway instance
  - Click the `Connect` button
  - On the `EC2 Instance Connect` tab keep the `Connect using EC2 Instance Connect` option selected
    - Click `Connect`
    - In the pop-up terminal you should be able to query the two interfaces of the gateway (one in the public, one in the private subnet), you should be able to ping the client(s)
- And now, let the fun (with `iptables`) begin

Task 1: Connect Client to the internet

Use your Gateway

- configure NAT (MASQUERADE)
  - with `iptables`
- enable routing in the GW
  - `ip_forward` kernel parameter. . .
- at the Client
  - define the default gw

Test (@Client)

- install the `ntpdate` package
  - `sudo apt update`
  - `sudo apt install ntpdate`

Task 2: Configure the firewall

```bash
#!/bin/bash
# delete chains
iptables -F FORWARD
iptables -X  # delete all user-specified chains
iptables -Z  # reset counters
# set default policies
iptables -P FORWARD DROP
# allow icmp traffic
iptables -A FORWARD -p icmp -j ACCEPT
# enable outgoing traffic
iptables -A FORWARD -s 10.0.0.0/24 -j ACCEPT
# enable backward direction if it was initiated from the internal domain
iptables -A FORWARD -d 10.0.0.0/24 -p tcp \
    -m state --state ESTABLISHED,RELATED -j ACCEPT
# enable DNAT ports from the external net
iptables -A FORWARD ! -s 10.0.0.0/24 -p tcp --dport 80 \
    -m state --state NEW -j ACCEPT
iptables -A FORWARD ! -s 10.0.0.0/24 -p tcp --dport 22 \
    -m state --state NEW -j ACCEPT
# enable DNS
iptables -A FORWARD -p udp --sport 53 -j ACCEPT
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
# log dropped packets
iptables -A FORWARD -m limit --limit-burst 5 --limit 2/s \
    -j LOG --log-prefix 'FIREWALL: ' --log-level 7
```

@Gateway

- configure the FORWARD chain of iptable's filter table
- default policy: DROP or REJECT
  - adapt the example config
    - IP addresses should be updated!
  - create a shell script (copy the source e.g. to nano editor)

Task 2: Configure the firewall

```bash
#!/bin/bash
# delete chains
iptables -F FORWARD
iptables -X  # delete all user-specified chains
iptables -Z  # reset counters
# set default policies
iptables -P FORWARD DROP
# allow icmp traffic
iptables -A FORWARD -p icmp -j ACCEPT
# enable outgoing traffic
iptables -A FORWARD -s 10.0.0.0/24 -j ACCEPT
# enable backward direction if it was initiated from the internal domain
iptables -A FORWARD -d 10.0.0.0/24 -p tcp \
    -m state --state ESTABLISHED,RELATED -j ACCEPT
# enable DNAT ports from the external net
iptables -A FORWARD ! -s 10.0.0.0/24 -p tcp --dport 80 \
    -m state --state NEW -j ACCEPT
iptables -A FORWARD ! -s 10.0.0.0/24 -p tcp --dport 22 \
    -m state --state NEW -j ACCEPT
# enable DNS
iptables -A FORWARD -p udp --sport 53 -j ACCEPT
iptables -A FORWARD -p udp --dport 53 -j ACCEPT
# log dropped packets
iptables -A FORWARD -m limit --limit-burst 5 --limit 2/s \
    -j LOG --log-prefix 'FIREWALL: ' --log-level 7
```

@Client

- make ntpdate work
- `sudo ntpdate ntp.ubuntu.com`
  - should sync the date/time
- (this also requires config @Gateway)

Task 3: Configure port forwarding

Make Client's webserver available from the inernet

- install apache2 webserver @Client
  - `sudo apt install apache2`
  - use the default port (80)
- configure DNAT @Gateway
  - making use of `iptables`
  - "open" tcp port 8080
    - externally we should access Client's webserver
    - via publicIP:8080 or publicDNS:8080
- update the security group @AWS accordingly

Optionally

- ssh can also be used
- (it does not require apache2)

IMSc Task: Enable load balancing

Webservers

- Set up 2 Client VMs as webservers
  - install apache2

@Gateway

- configure DNAT rules into iptables
- configure load balancing
  - to share the load between the 2 webservers

Testing

- validate the operation
- use different tools for testing

Infrastructure cleanup using CloudFormation (mostly)

- Log out of the instances
- In the EC2 console click `Instances` and select the gateway instance (checkmark it)
  - Click the `Actions` button (top right) then select the `Networking` → `Detach network interface`
  - Under `Network interface` select the one with the name `gateway-private-int`
  - Click the `Detach` button
    - Explanation: CloudFormation cannot detach the gateway's private interface because it did not attach it
  - Look for the Private IPv4 addresses of the gateway instance under its Details tab
  - Wait until the `10.0.128.5` IP address disappears and only `10.0.0.5` is left (you might need to use the EC2 console's refresh button for this change to show up)
- Go to the CloudFormation console
  - Select the `vitmac12` stack
  - Click the `Delete` button (top right)
  - Click the `Delete` button again to confirm your intention
  - The stack will be deleted in a few minutes:
    - First it will enter the `DELETE_IN_PROGRESS` then the `DELETE_COMPLETE` state, after which the stack will disappear from the list of active stacks
    - There will be one stack left with the description `associate Learner Lab template (academy)`
    - You find your deleted stack if under `Filter status` you select `Deleted` instead of `Active`

AWS Academy

- AWS Academy Cloud Foundations
  - Module 10 - Auto Scaling and Monitoring
    - Lab - 6 Scale & Load Balance your Architecture