**_COMPX527 Week 6-Lecture 1 & 2_**

# Cloud Architectural Concepts

## Agenda

- Announcements
- Application Architecture
- Pillars of Least Privilege in Cloud
- Secure Application Architecture and Infrastructure as Code
- Terraform

## *Announcements*

1. Group Project - Final **Group** Submission
   - ***Deadline: 30th September 2025 at 9:00 am.***
   - Final Report
   - Video link (Presentation + demo)
   - GitHub link
     - Repository name = ***your project name***
     - **Private** repository
     - Add **me and the tutors** as collaborators before the deadline so we can access your repo.

## Announcements (Contd)

- Repo must contain:
  - Application source code
  - Automation instructions and deployment script(s)
  - README.md explaining how to set up and run your application.

- ***Important Notes***
  - The last commit **before the deadline** will be treated as the final submission (-10 marks reduction for any later changes in the commit)
  - If your repo is not accessible at the time of review, it will be marked as **not submitted**.
  - **Do not delete** the repo until grades are released.

## *Announcements (Contd)*

### 2. Peer Evaluation/Assessment – Individual Submission

- ***Deadline: 30th September 2025 at 9:00 am.***

### 3. Presentation Days – S.1.04

- 1PM - 4PM (Thursday, 2 October)
- 11 AM - 5PM (Friday, 3 October)

### 4. In-Class Test

- 9th October 2025
- Topics Covered from Week 1 - Week 10, including Legal and Compliance (excluding weeks 7 and 8)

# *Video Presentation - Guidelines*

- Make eye contact with the camera.
- Use bullet points on slides, avoid full sentences.
- Avoid reading directly from the screen.
- Use emphasis and pauses to highlight important points.
- Clear voice, no background noise
- Speak clearly and at a steady pace.
- Video must be **7 minutes long**, and **every member** must present their tasks.
- Project should be in **running state** during the presentation day.

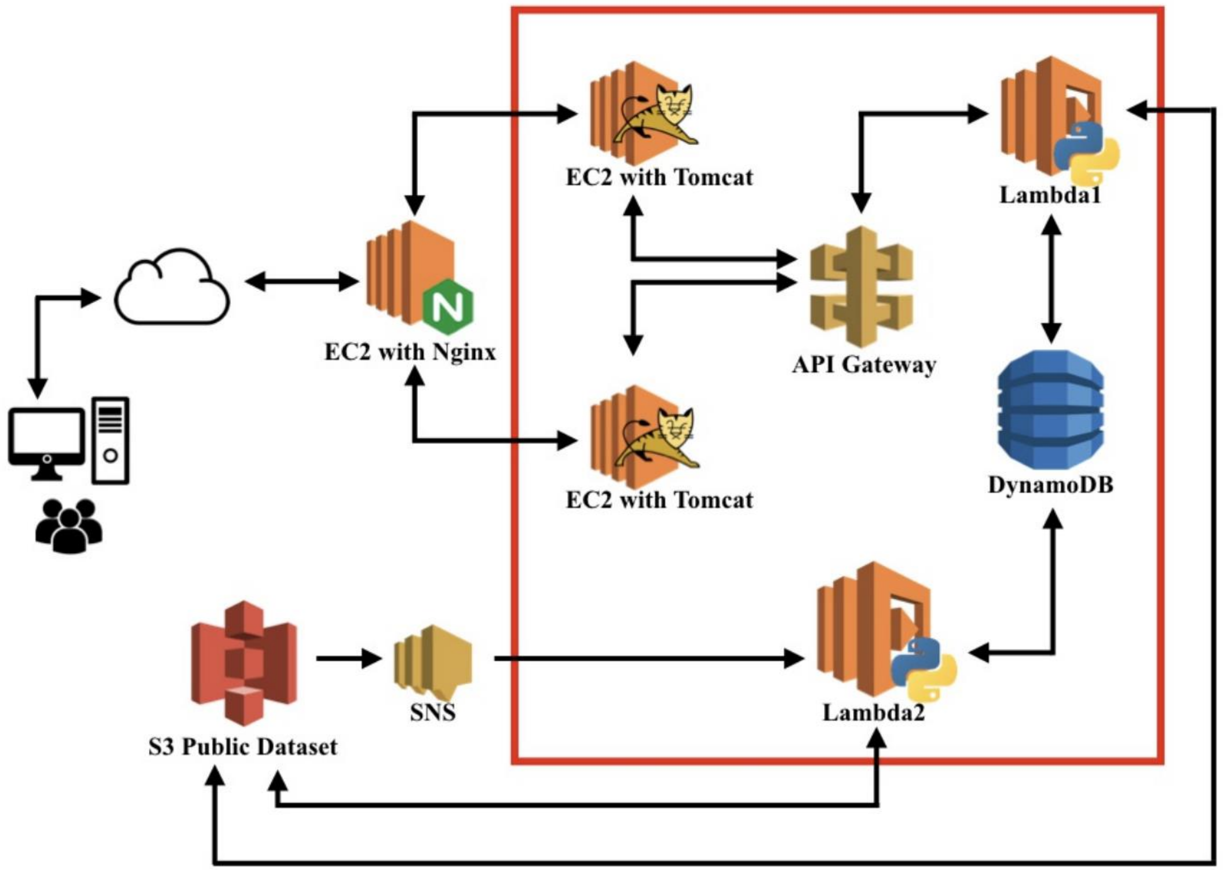
## Reminder

- Revised Projects
  - Email by tomorrow
- Citation and References

## Group Project

- Team communication
  - **Meeting minutes along with screenshots**
  - **Include tasks done last week, current week, and future week**
- Limit Your Expenditure
  - Monitor your services
    - Use the AWS budget service
    - Set up CloudWatch alarms
    - Only switch the services on when being used
    - Do not leave services on for long periods of time
- Start thinking about your architecture

# *Application Architecture*

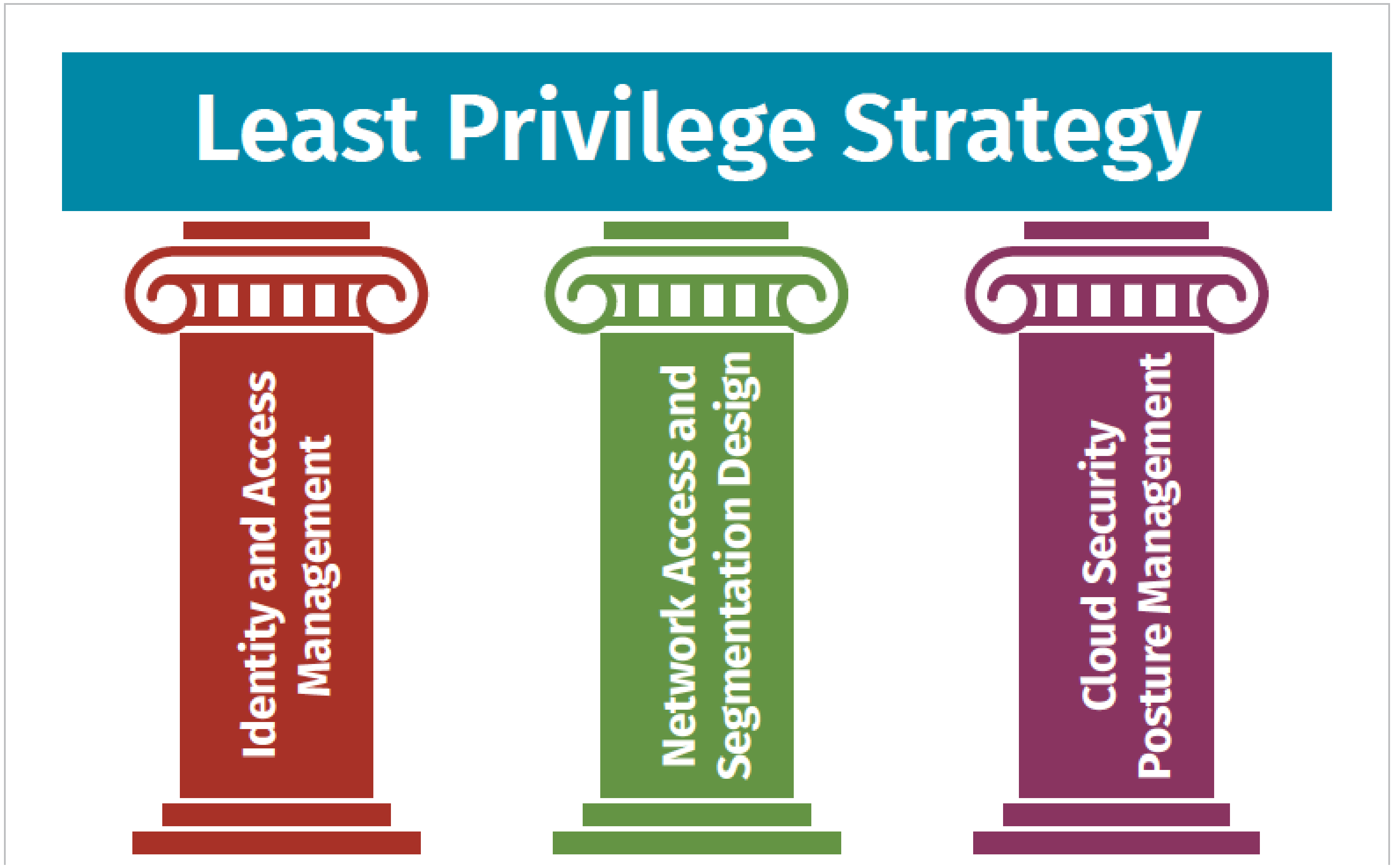


## Least Privilege in the cloud- A challenge

## *Challenges in implementing least privilege*

- The ability to determine the appropriate "least privilege" for a given use case is a surprisingly complex issue.
- Even successful least privilege implementations tend to shift and drift over time.
- Too strict access controls impact usability.
- Enforcing will be challenging in dynamic environments, adopting the **zero-trust principles** help address this issue.

## Pillars of Least Privilege in Cloud


Least Privilege Strategy
Identity and Access Management
Network Access and Segmentation Design
Cloud Security Posture Management

## IAM

- Who needs high privileges:
  - Privileged users - administration, engineering, and security-focused tasks.
  - Deployment pipelines and associated systems and services.
- Relationship Mapping
  - Map cloud user and service relationships to create the most restrictive privilege models needed.
  - Tools such as Trusted Advisor etc.

## Network Segmentation

- Lateral movement issue
- A least privilege model also reduces the scope of **impact** when an attacker has illicitly gained access to an asset within a cloud environment.
- The classic model for implementing least privilege at the network level starts with a network access control policy of **Deny All** and then adds only those types of network access needed.
- Example: VPC

## *Cloud Security Posture Management (CSPM)*

- Monitoring for your services, accounts, and privileges for known misconfiguration issues.
- Example: Prowler, AWS has Security Hub, third-party CSPM tools

## Secure Application Architecture and Infrastructure as Code (IaC)

## Scenario Exercise

- Assume you are part of a DevOps team for an organisation. You are responsible for:
  - – Provisioning infrastructure for critical web applications on the cloud
  - – Deploying the applications
  - – Applying security patches to the infrastructure and applications
  - – Providing 24/7 application and infrastructure support

## Scenario Exercise

- Small/medium/large infrastructure
- Doing these tasks manually:
  - Same tasks are repeated multiple times
    - Error-prone
    - Task fatigue
  - Inconsistent configuration and state across your infrastructure
- Automation, Orchestration and Configuration management

## *Automation, Orchestration, Configuration Management*

- Automation:
  - Using tools and software to perform repeatable configuration actions and processes.
  - Automating a single task: creating an EC2 instance
- Orchestration
  - Coordinating multiple automated tasks to work together in a sequence or in parallel.
  - Example: deploy an application on an instance.
- Configuration Management:
  - Keeps your configuration uniform across your infrastructure

# *Infrastructure as Code (IaC)*

- IaC enables automation, orchestration, and consistent configuration management at scale.

AWS CloudFormation

Terraform

aws CDK

Terragrunt

OpenTofu

Pulumi

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

# *TERRAFORM*

## *Infrastructure as Code*

Presented by: Shashwot Risal

Source: https://developer.hashicorp.com/terraform/intro

## *Topics*

- What is a Terraform?
- Working of Terraform
- Manage any infrastructure
- Track your infrastructure
- Automate changes
- Standardize configurations
- Collaborate
- DEMO

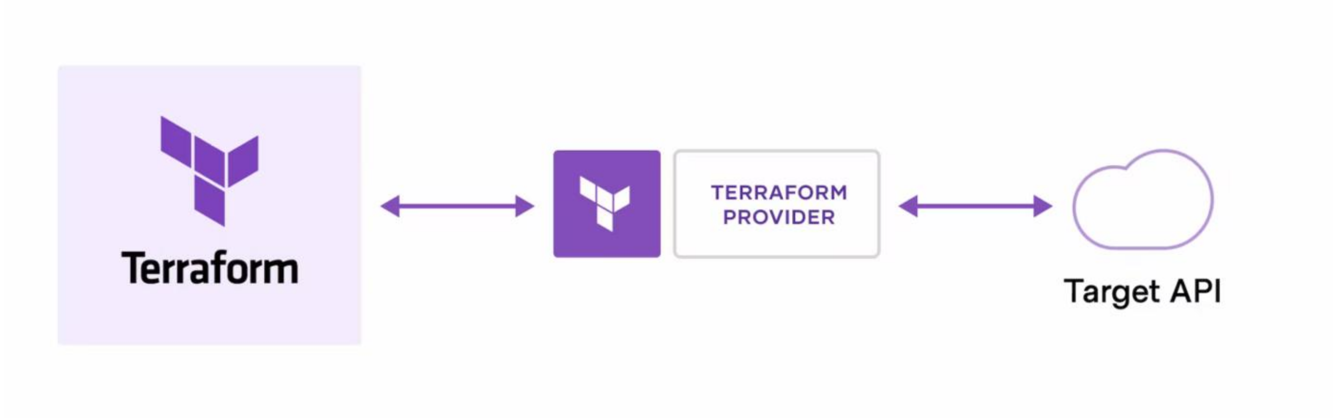
Source: https://developer.hashicorp.com/terraform/intro

## What is Terraform ?

- Tool by Hashicorp (IBM)
- It lets us define (both cloud and on-prem resources) **human-readable configuration files** that we can version, reuse, and share.
- It can manage **low-level components** like compute, storage, and networking resources as well as high-level components like DNS entries and SaaS features.

## *Working of Terraform*

- Terraform **creates and manages** the resources on cloud platforms and other services through the **API**.

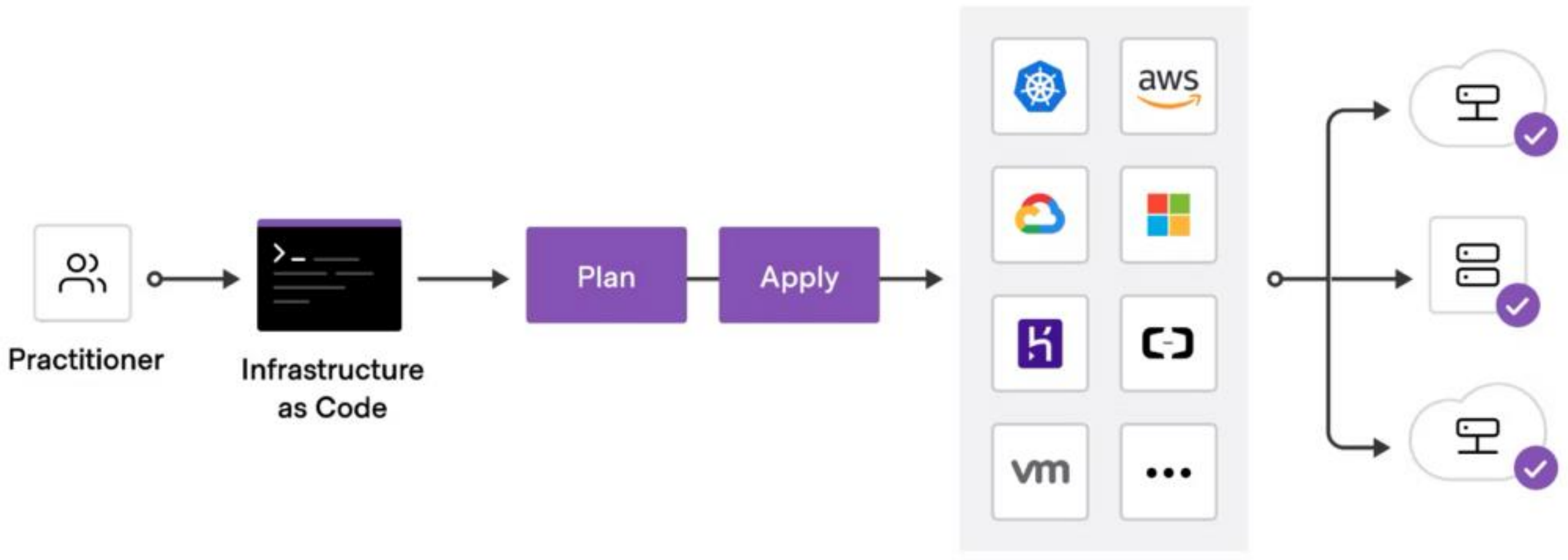


https://developer.hashicorp.com/terraform/intro

## Working of Terraform

- Write
  - **Define resources**, which may be across multiple cloud providers and services.
  - E.g., create an EC2 instance, create a VPC.
- Plan
  - **Create an execution plan** describing the infrastructure it will create, update, or destroy based on the existing infrastructure and configuration.
- Apply
  - On **approval**, Terraform performs the proposed operations in the correct order.
  - For example, if you scale your EC2 instances, Terraform will recreate the autoscaling group and update your resources.

## How does Terraform work?



https://developer.hashicorp.com/terraform/intro

## Manage any infrastructure

- Find providers for many of the platforms and services you already use in the **Terraform registry**.
- You can also **write your own code**.
- Terraform takes an **immutable approach**, reducing the complexity of upgrading or modifying your services and infrastructure.
    - Mutable approach: Stop the instance, change its type, start it again.
        - Risks: downtime, unexpected settings still hanging around.
    - Immutable approach (Terraform): Create a new t3.micro instance, update references (like load balancer targets), then destroy the old t2.micro.

## Track your infrastructure

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

✓ Terraform generates a plan

Prompts for approval before modifying an infrastructure.

It keeps track of a real infrastructure in a state file, which acts as a source of truth for the environment.

Terraform uses a state file to determine the changes to make to your infrastructure so that it will match your configurations.

## *Automate Changes*

- Terraform configuration files are **declarative**
  - Describe the end state of the infrastructure.
- Terraform handles the underlying logic.
- Terraform builds a resource graph to determine the resource dependencies
- Terraform creates or modifies non-dependent resources in parallel.

## Standardise Configurations

- Terraform supports reusable configuration components called **modules** that define configurable collections of infrastructure.
- Publicly available modules can be used from the Terraform registry, or you can write your own.

## Collaborate

- Since a configuration is written in a file,
  - It can be committed to a Version Control System (VCS)
- Use **Terraform Cloud** to efficiently manage the Terraform workflow across teams.

THE UNIVERSITY OF WAIKATO
Te Whare Wānanga o Waikato

***DEMO***