Learn Infrastructure-as-Code through Minecraft

Stop Punching Trees and Start Mining for Diamonds!!!

Hello
my name is

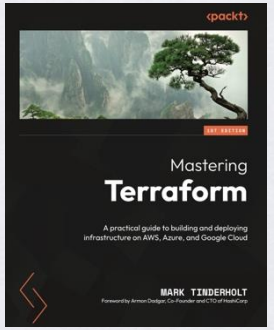
Mark Tinderholt

Principal Architect

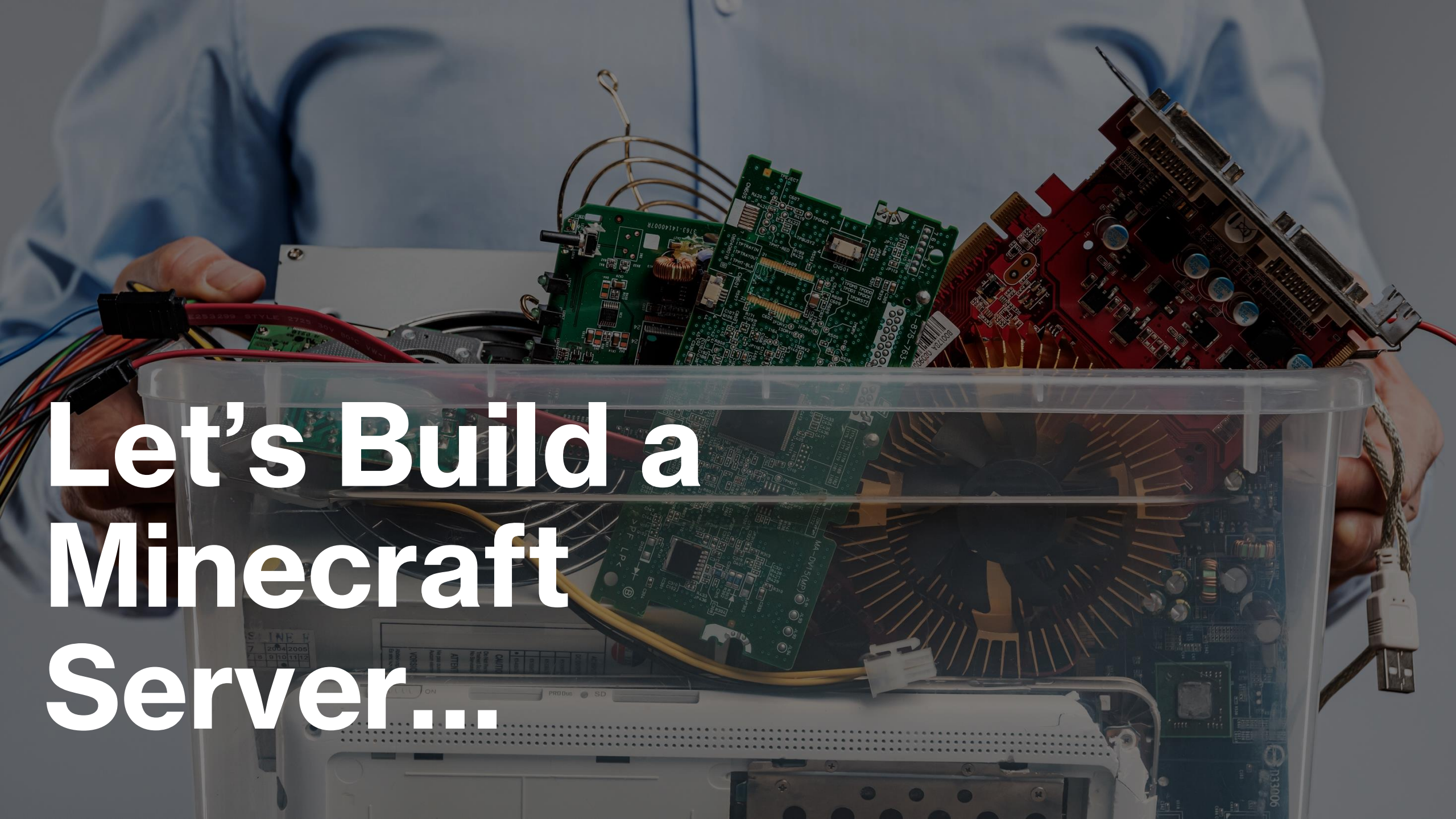
Microsoft

Microsoft Azure

HashiCorp
an IBM Company
AMBASSADOR
2025




Ready to
become
Super DAD?

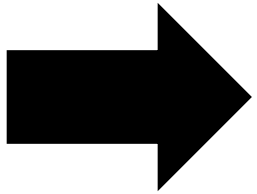


Let’s Build a Minecraft Server...


...and Learn
Infrastructure-
as-Code!!!

What is Infrastructure-as-Code?

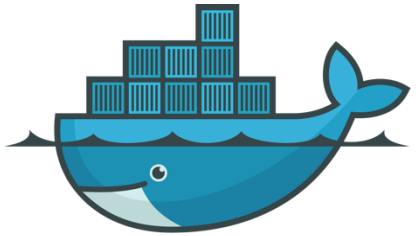
Infrastructure-as-Code is writing and managing infrastructure in the same way we write and manage application code.



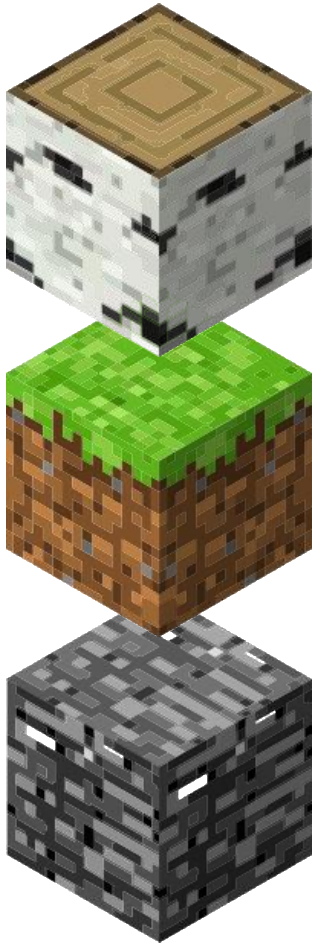
Code

Application Environments

The Cloud Crafting Table...



Configuration Spans Many Layers...

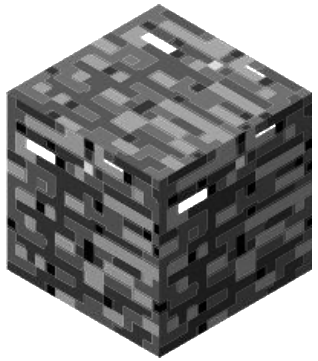


Application Configuration

Operating System Configuration

Cloud Services Configuration

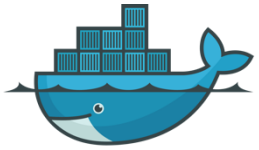
Cloud Services Configuration



Definition: The foundational cloud services and resources your application runs on.

- Virtual Networks (VNets), Subnets, and Network Security Groups
- Virtual Machines (VMs)
- Azure Kubernetes Service (AKS) clusters
- Platform-as-a-Service (PaaS) offerings (App Service Plans, Azure SQL, Azure Functions)
- Serverless compute services

Operating System Configuration



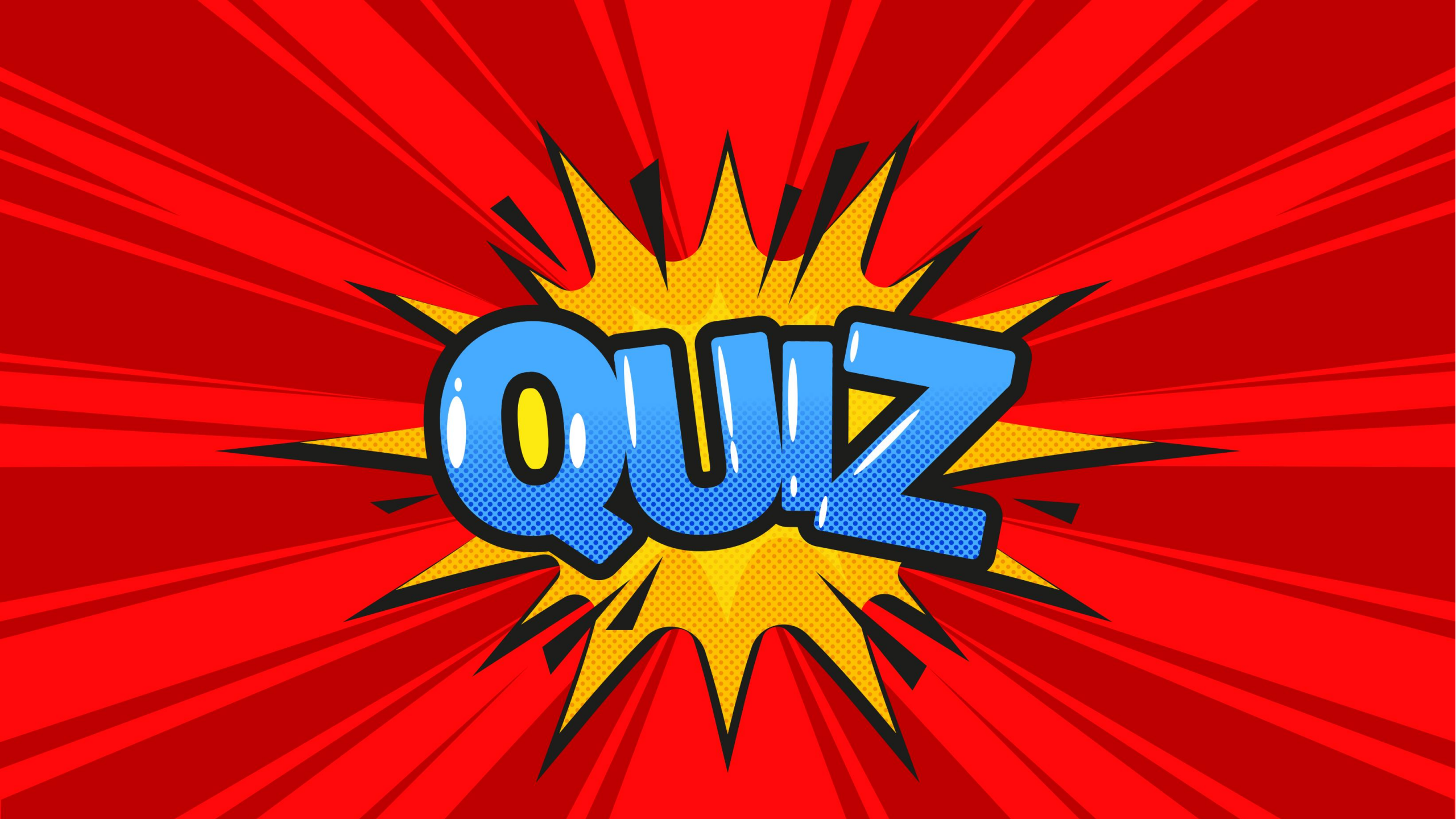
Definition: How the server (physical or virtual) or container is configured at the OS level.

- OS selection and version (e.g., Ubuntu 22.04 LTS, Windows Server 2022)
- Installed software, libraries, and runtimes (Java, .NET, Python, Node.js)
- Open ports, firewall rules, and listening services
- System packages and updates
- Application installation scripts and configuration files
- Creating admin user accounts or roles
- Base images built with **Packer** or container images built with **Docker**

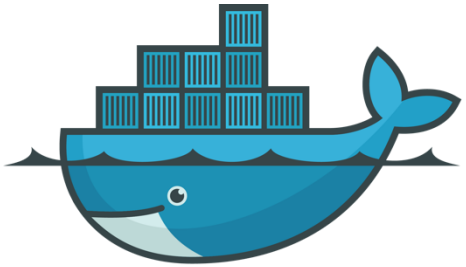
Application Configuration

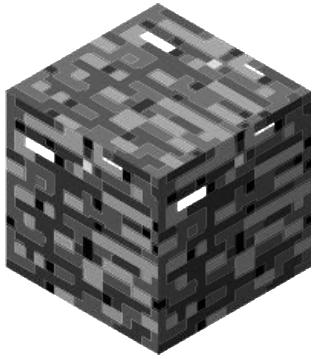
Definition: The configuration that happens *after* the application runtime environment is provisioned, defining how the application is deployed, connected, and behaves.

- Workload deployment definitions (services, deployments, namespaces, routing rules)
- Environment variables (API keys, connection strings)
- Seed data for databases
- Feature flags
- Tenant onboarding or provisioning automation
- Loading default templates, content, or assets
- Application-specific tuning (e.g., caching settings, queue thresholds)


QUIZ

Guess Who #1???



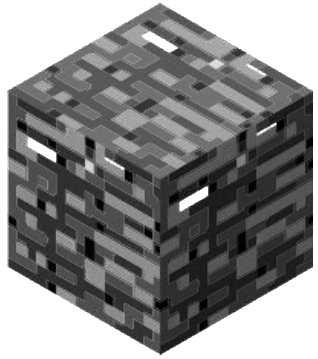


Cloud Services

Operating System

Application

Guess Who #2???

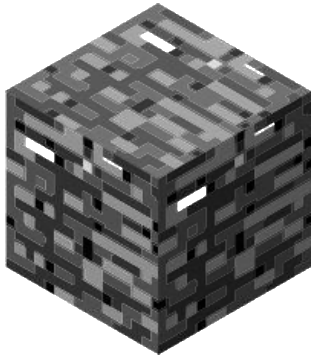


Cloud Services

Operating System

Application

Guess Who #3???



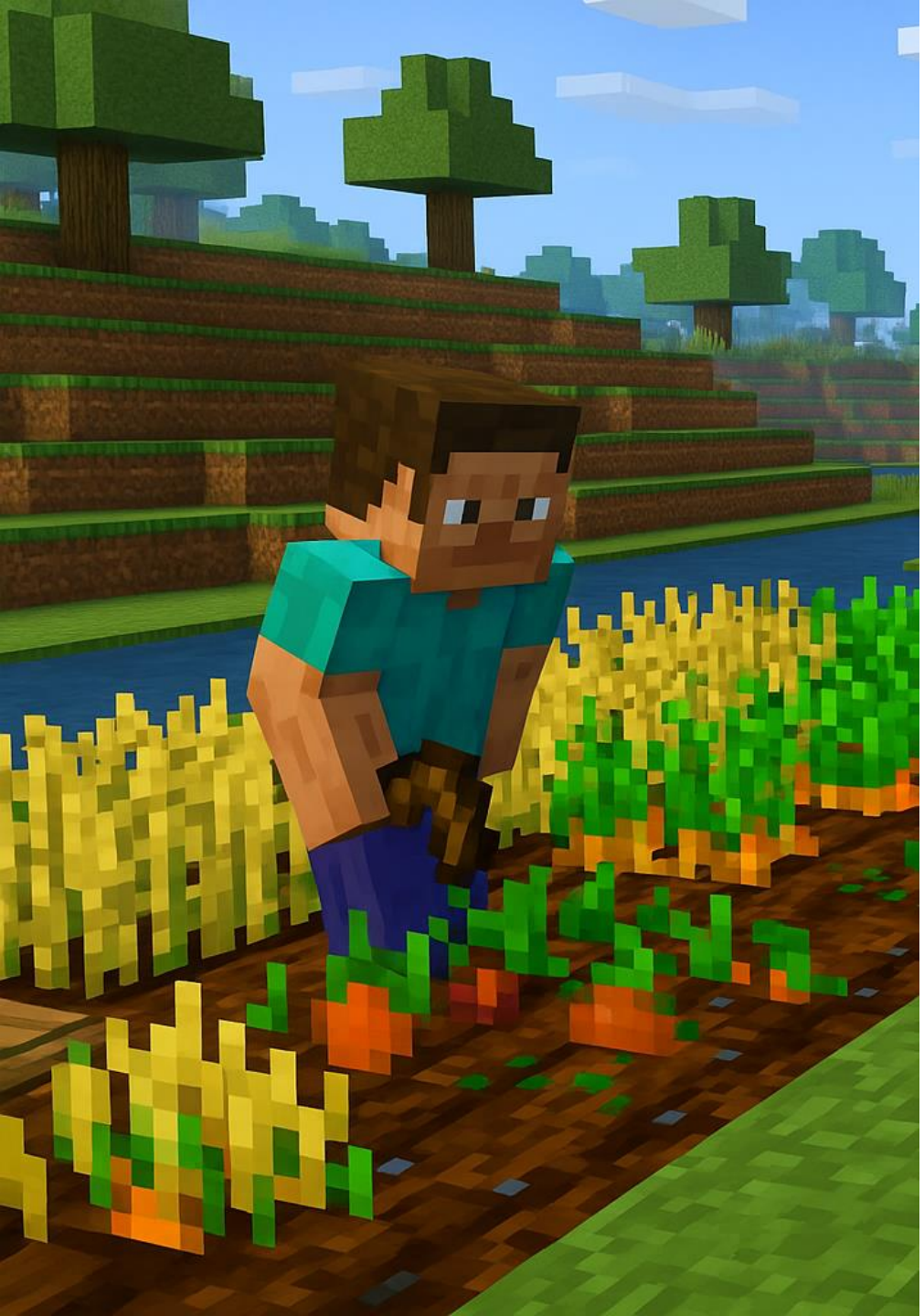
Cloud Services

Operating System

Application


Roles &
Responsibilities

Crafting



Platform Teams

Focus: Application-agnostic infrastructure

- Provision and manage foundational cloud resources (Networks, firewalls, VPN)
- Build and maintain shared compute platforms (Kubernetes clusters, shared compute node pools)
- Set up centralized logging, monitoring, and alerting (Log Analytics, Azure Monitor, Prometheus, Grafana)
- Implement identity, access controls (Azure AD, RBAC)
- Enforce compliance and security baselines (policies, guardrails)
- Publish reusable IaC modules and templates for app teams to consume

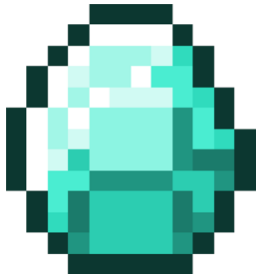
App Dev Teams

Focus: Application-specific infrastructure

- Provision application hosting resources (web apps, function apps, container instances)
- Deploy workload-specific services (databases, message queues, API gateways)
- Manage networking for their app (custom domains, WAF rules, private endpoints)
- Define application / service models (manifests, Helm charts, container images)
- Configure application-specific monitoring, logging, and alerts
- Leverage shared modules from the platform team while customizing to their app's needs

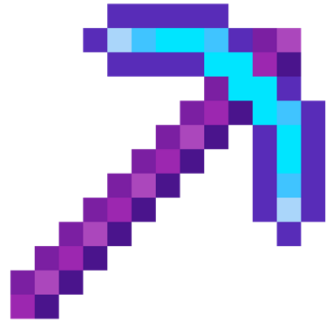

ENCHANTED PICKAXE
MINING ORE, CRAFTING TABLE,
SMELTING, ENCHANTING, XP FARMING

How to get an Enchanted Pickaxe...



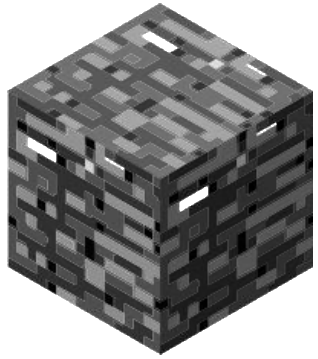
1

2



3

Terraforming Minecraft



Cloud Services

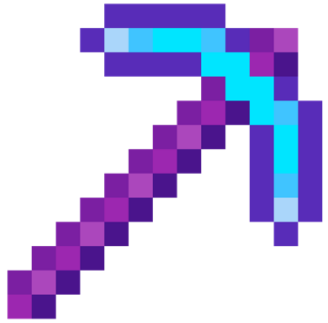
Operating System

Application

How to deploy to production...

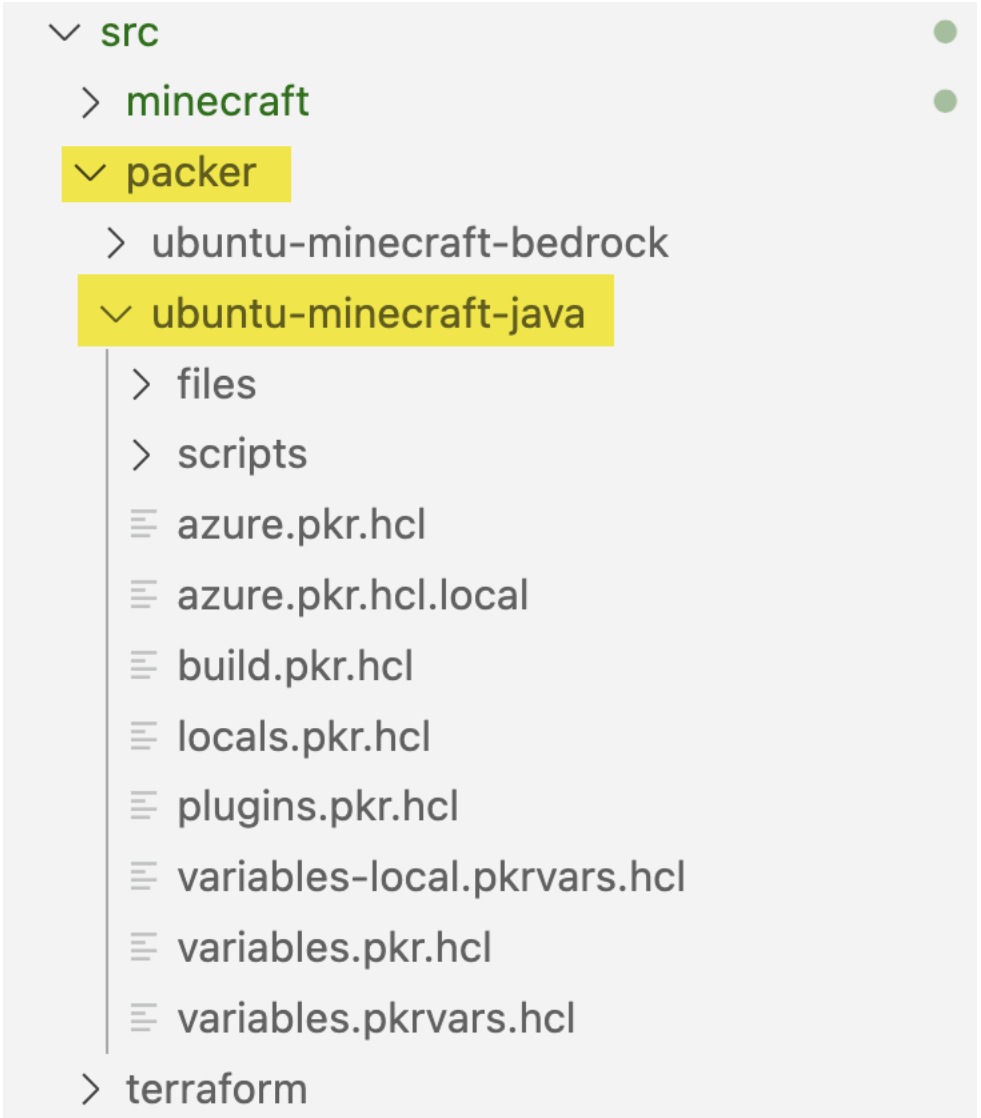
1

2



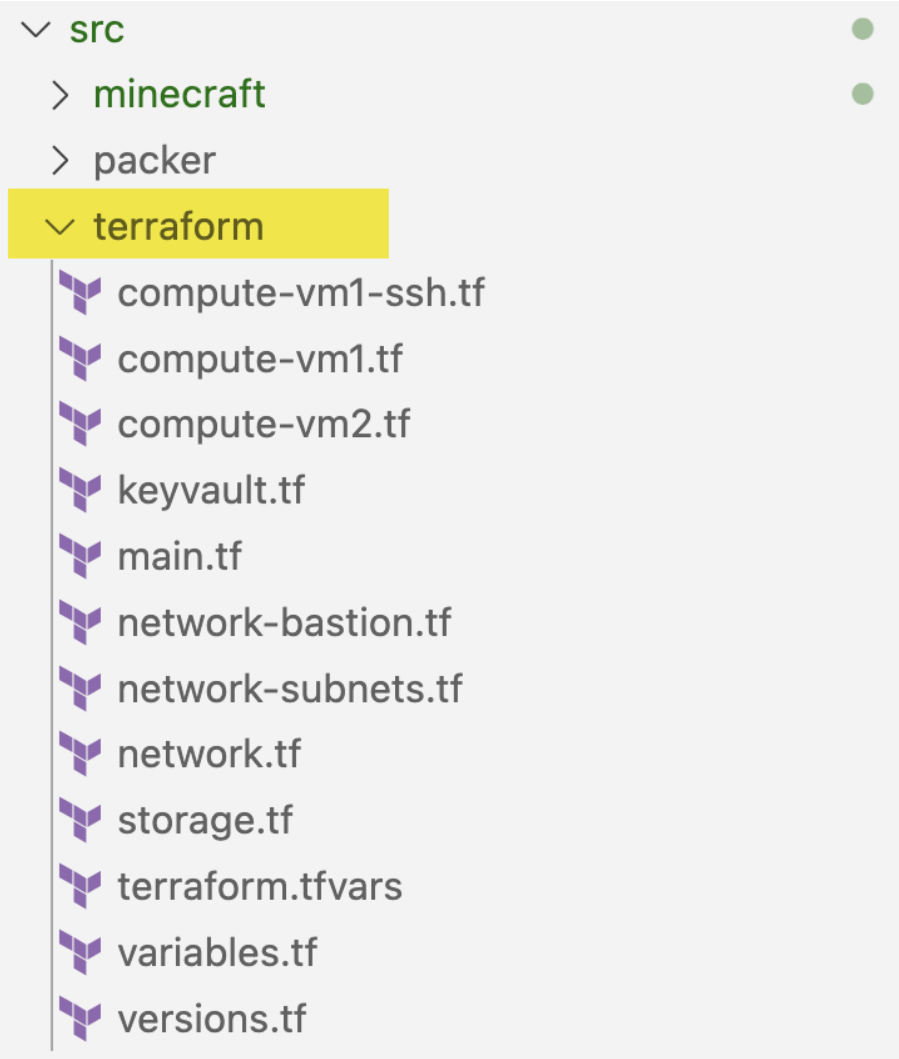
3

Packer Workspace



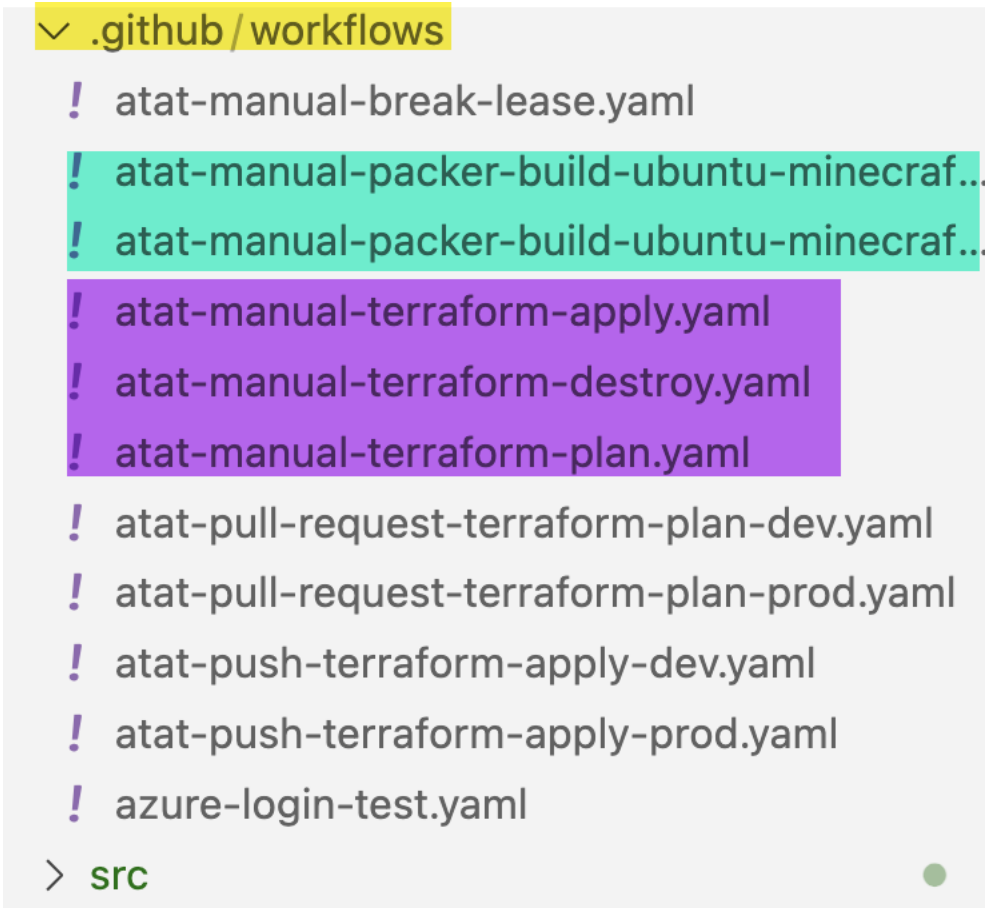
- **Connects to Azure:** Dynamically provisions a temporary VM where we can build our machine, shutdown and capture the image.
- **Install Software:** Azure CLI, curl, grep, zip/unzip, jq, OpenJDK
- **Setup a Service Account:** Home Directory and permissions for “mcserver” user
- **Install Minecraft:** Download & install latest package, setup Linux service, drop operational scripts (e.g. backup, restore, upgrade)

Terraform Workspace



- **Setup Network:** Virtual Network, Azure Bastion for Remote Access, NSG to allow Minecraft Ports
- **KeyVault:** Stores SSH Keys
- **Storage:** Backup & Recovery
- **VMs:** One Bedrock Server and One Java Server

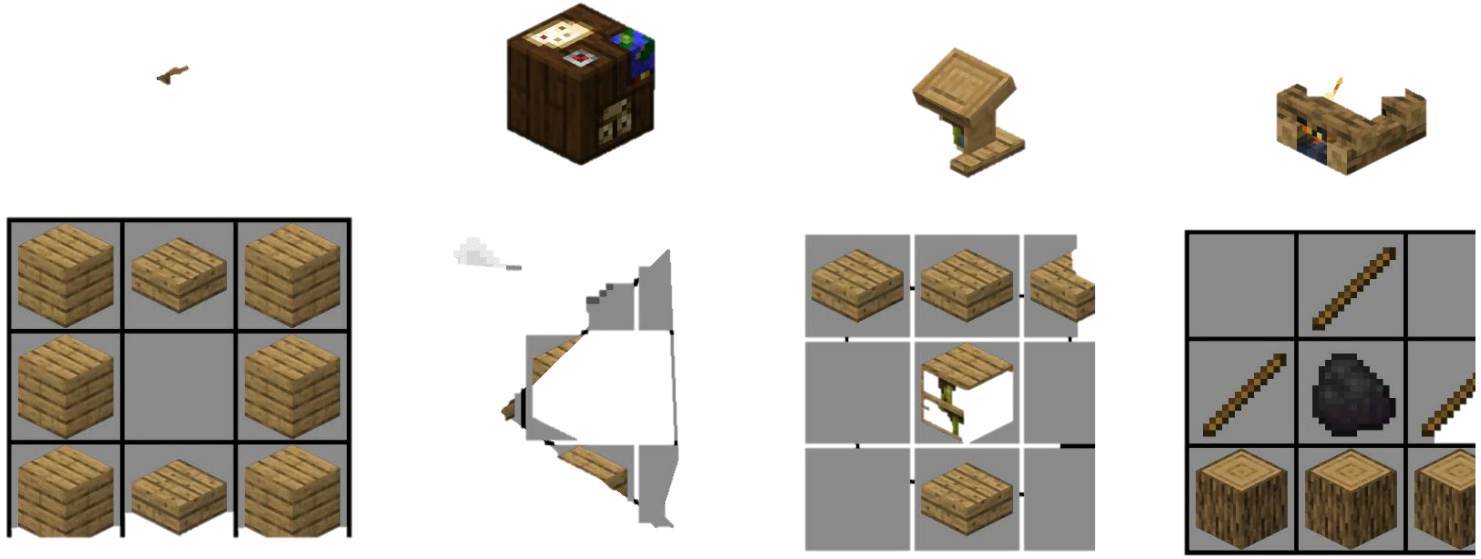
GitHub Actions – GitOps Workflow



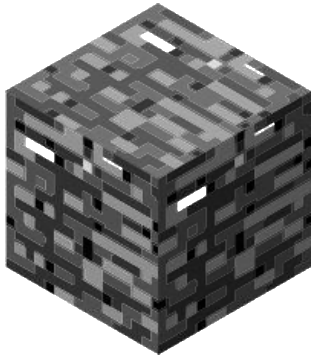
- **Packer Build:** One Bedrock Server and One Java Server
- **Terraform Core Workflow:** Plan, Apply, Destroy
- **Pull Request Process:** Run Plan before commit
- **Release Process:** Run Apply after commit for ‘develop’ and ‘main’


vmware
CITRIX
Akamai
DELL
Ping Identity
ORACLE
JFrog
aws
IBM
Redis
HELM
NetApp

Another Superpower: Modularity



Terraforming Minecraft

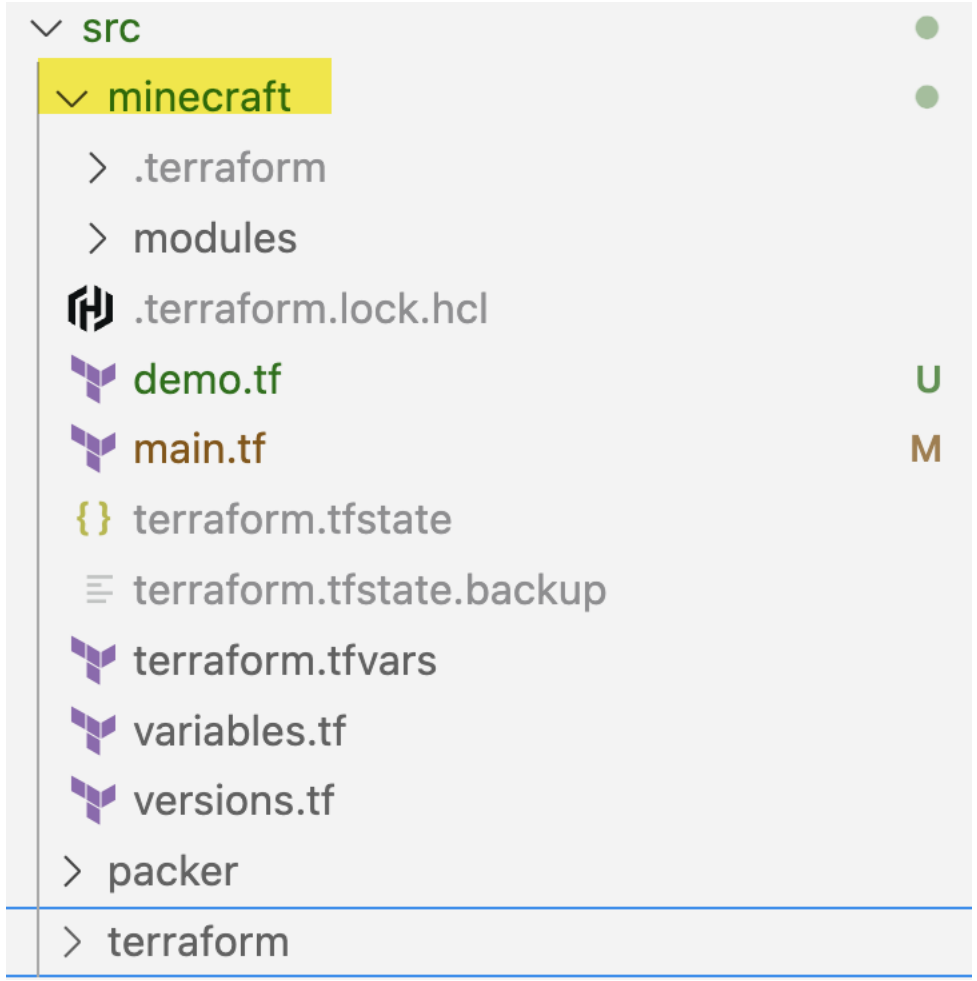


Cloud Services

Operating System

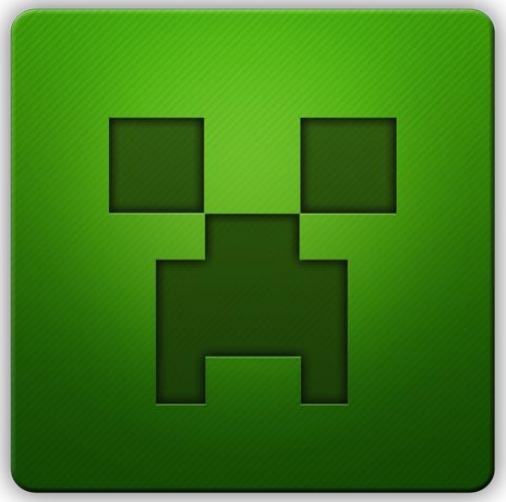
Application

Minecraft Workspace



- **Log Cabin**
- **Diamond Pyramid**
- **Golden Pyramid**
- **Iron Pyramid**
- **Powered Rail System**
- **Aqueduct**

The Minecraft Terraform Provider!!!



Minecraft Provider

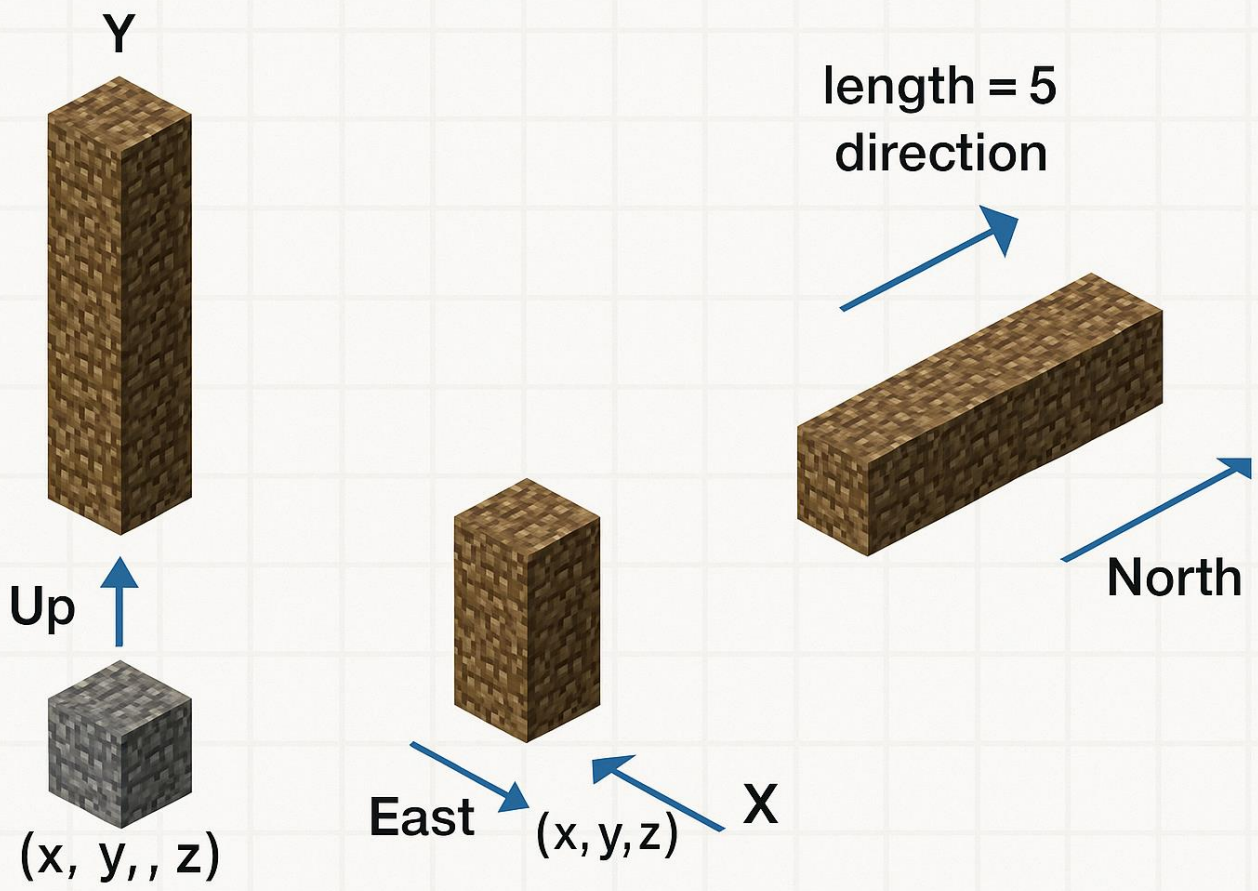
“minecraft_block”

Provider Primitives: The Block Resource

“minecraft_block”

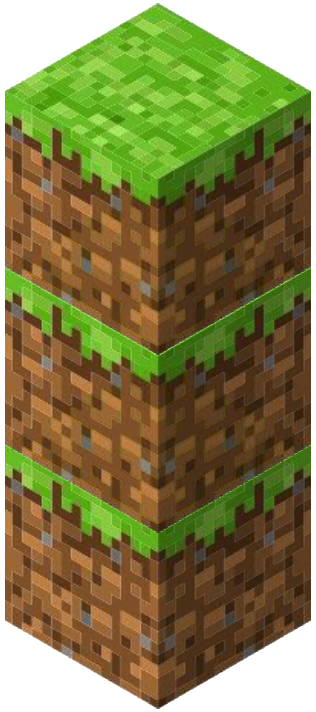
```
resource "minecraft_block" "dirt" {
  material = "minecraft:dirt"

  position = {
    x = -198
    y = 66
    z = -195
  }
}
```

Our First Module: The Pillar

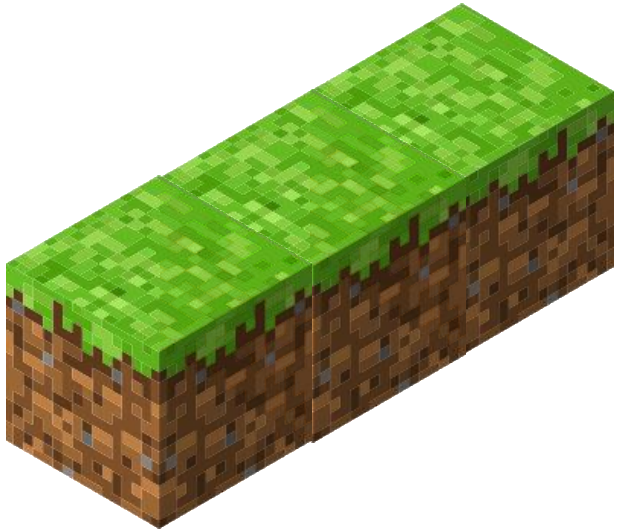



```
module "p1" {

  source = "./modules/pillar"

  material = "minecraft:dirt"
  length   = 3

  start_position = {
    x = -1532,
    y = 66,
    z = -1177
  }

}
```

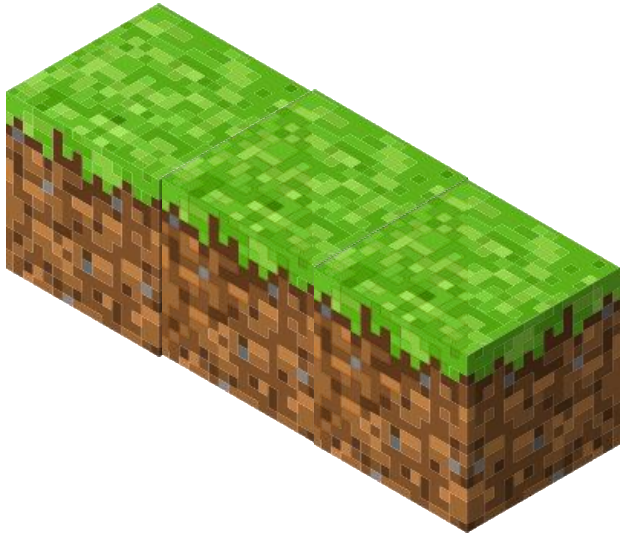
The Pillar in Action



Length: 3
Direction: Z

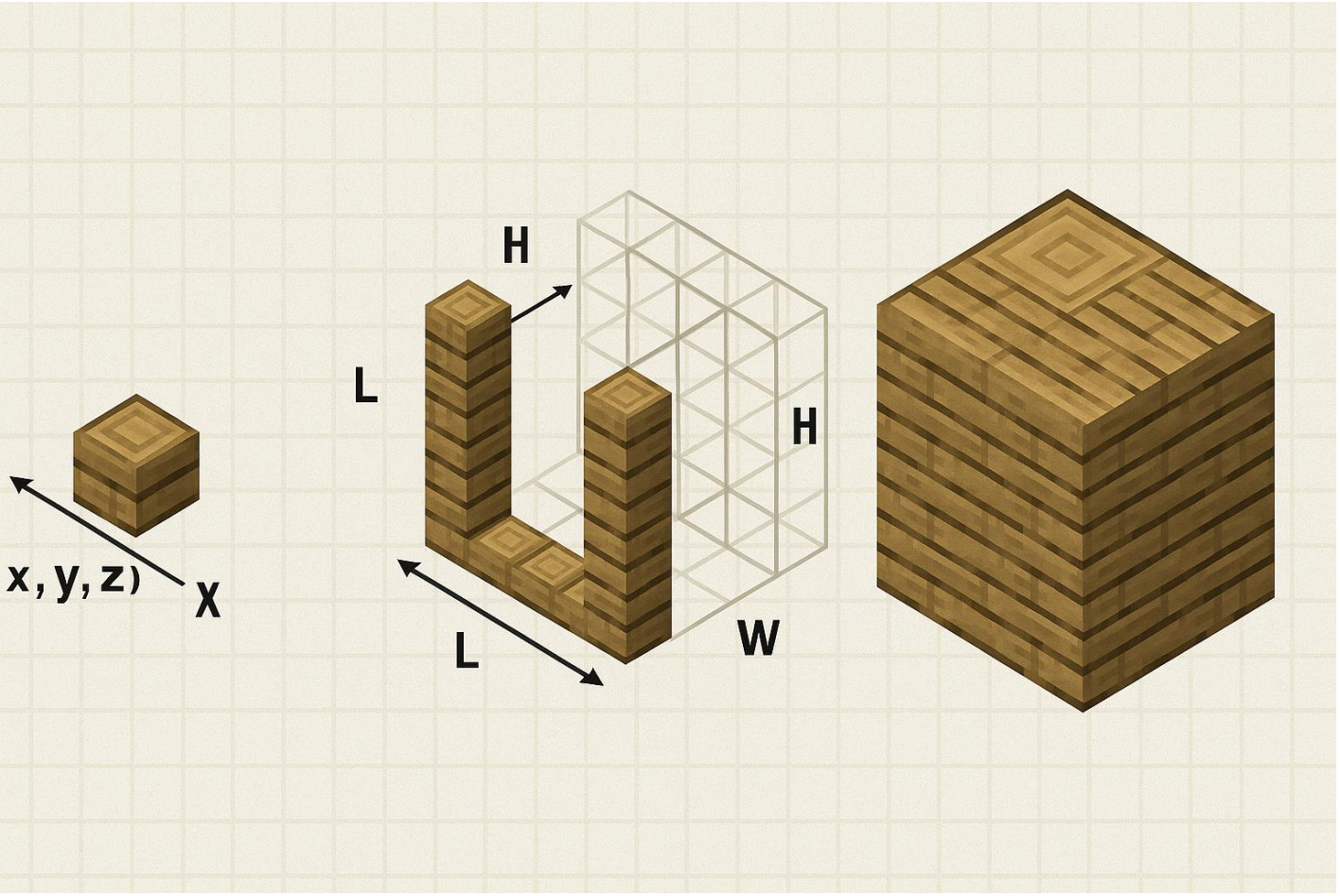


Length: 3
Direction: Y



Length: 3
Direction: X

Let’s Build Bigger: The Cuboid Module




```
module "island_near_nether_portal" {

  source = "./modules/cuboid"

  material = "minecraft:dirt"
  length   = 3
  width    = 8
  depth    = 8

  start_position = {
    x = -1564,
    y = 60,
    z = -1181
  }

}
```

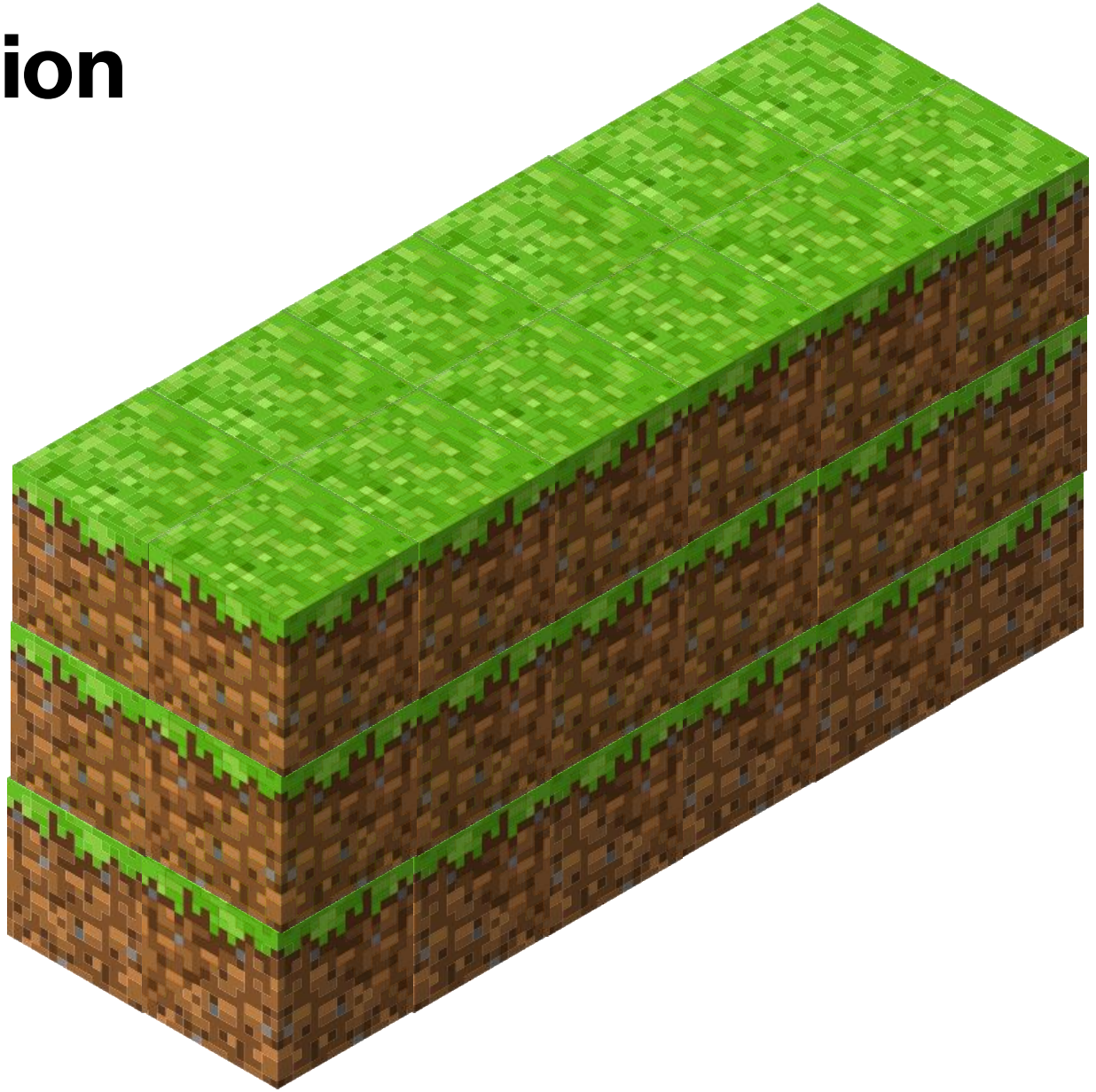
The Cuboid in Action

Length: 2

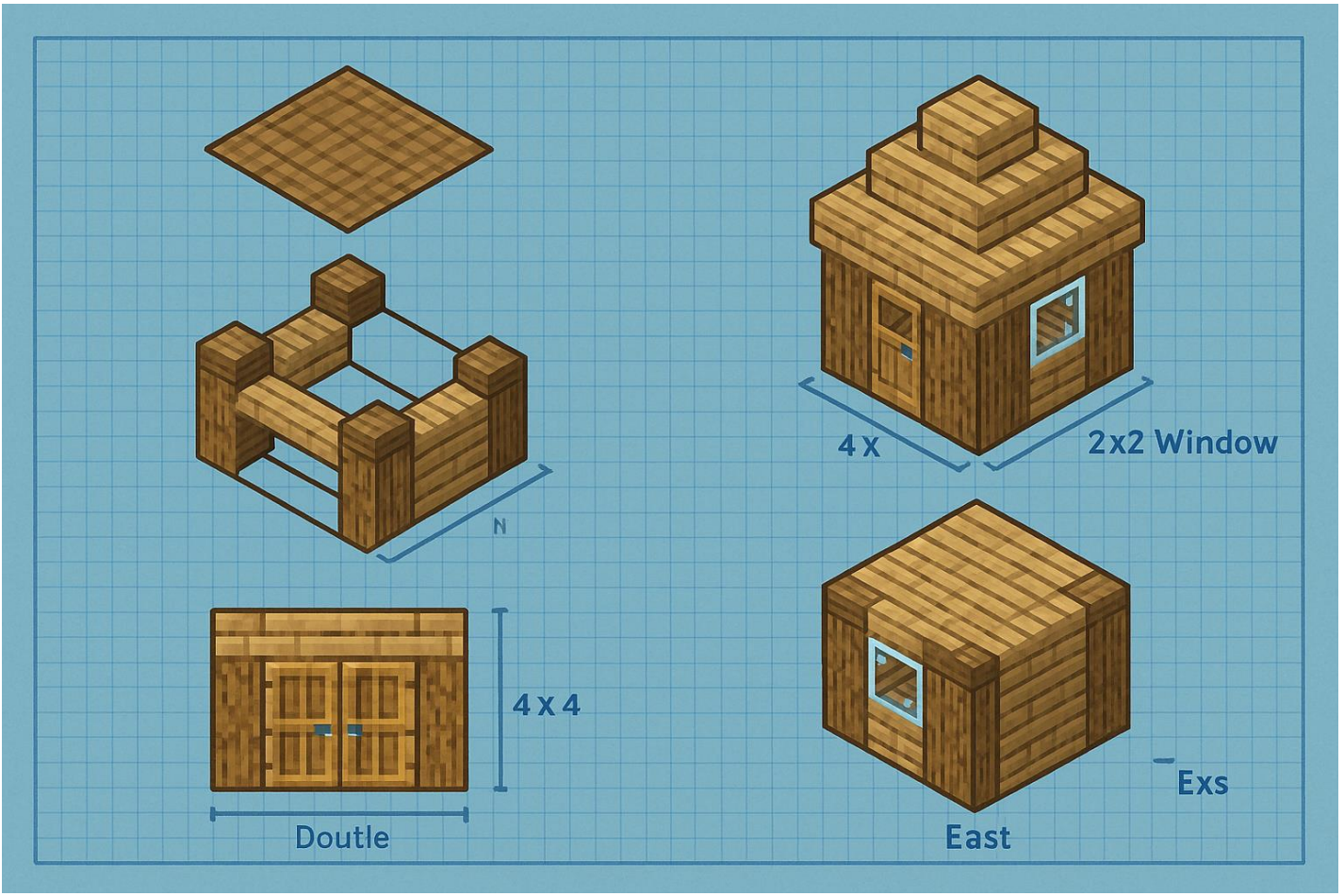
Height: 3

Depth: 8

Direction: Y

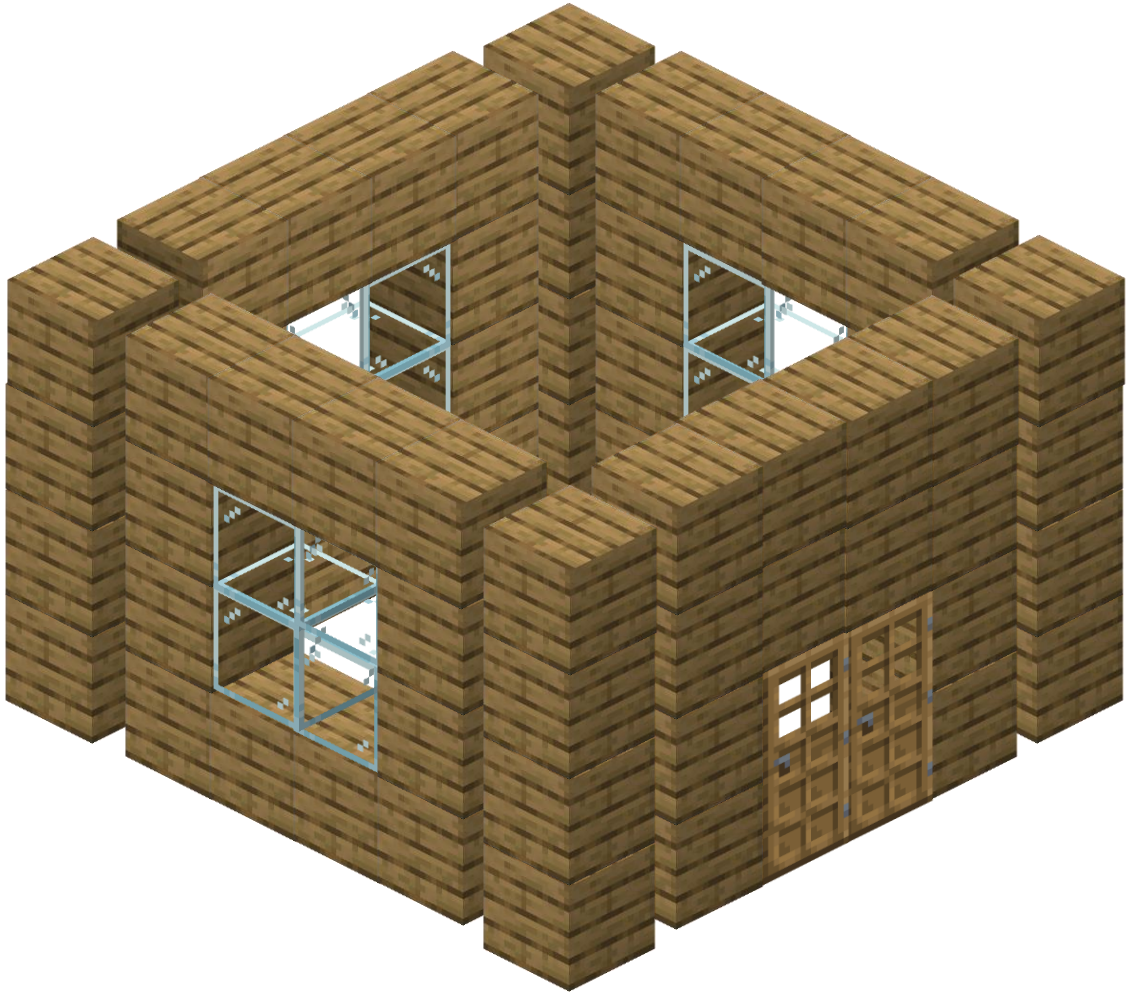


Log Cabin Module Design

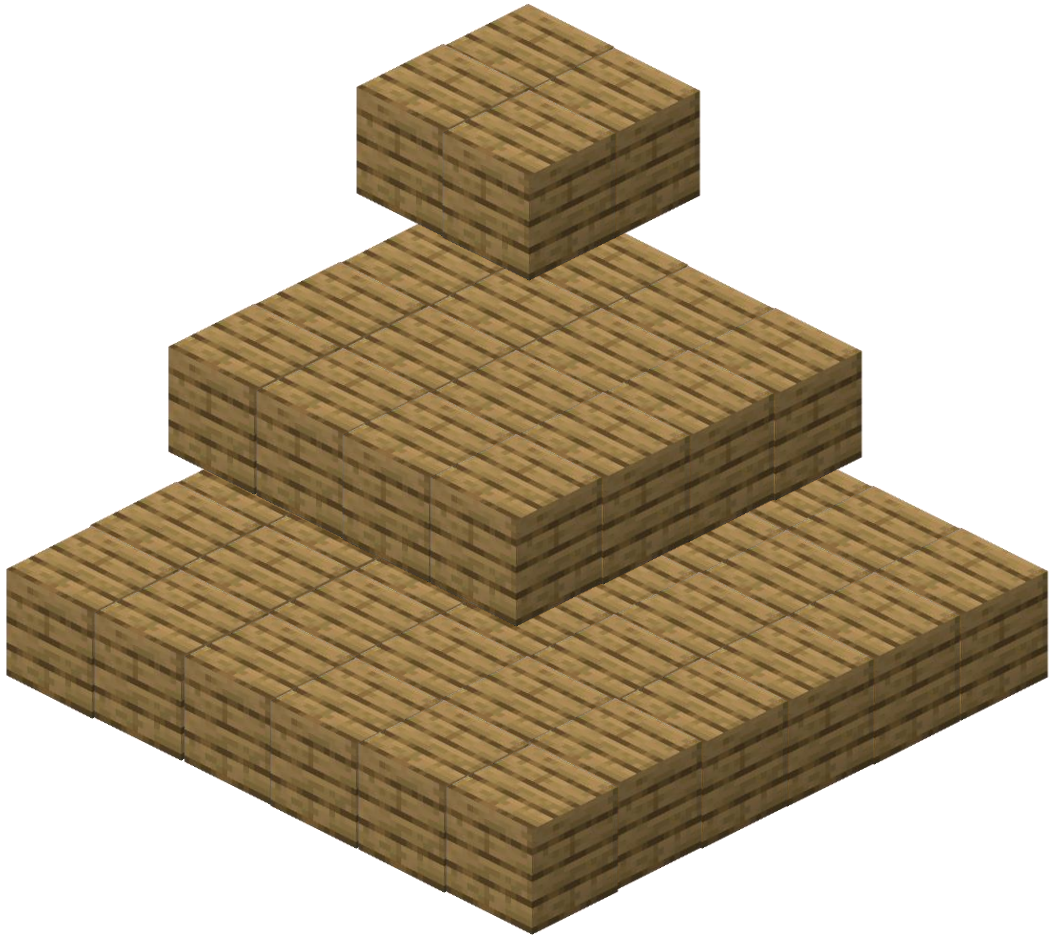



```
module "w1" {

  source = "./modules/small-hut"

  material        = "minecraft:oak_planks"
  glass_material = "minecraft:glass_pane"

  start_position = {
    x = -1562,
    y = 63,
    z = -1180
  }

}
```

Now, Let’s Build a Log Cabin

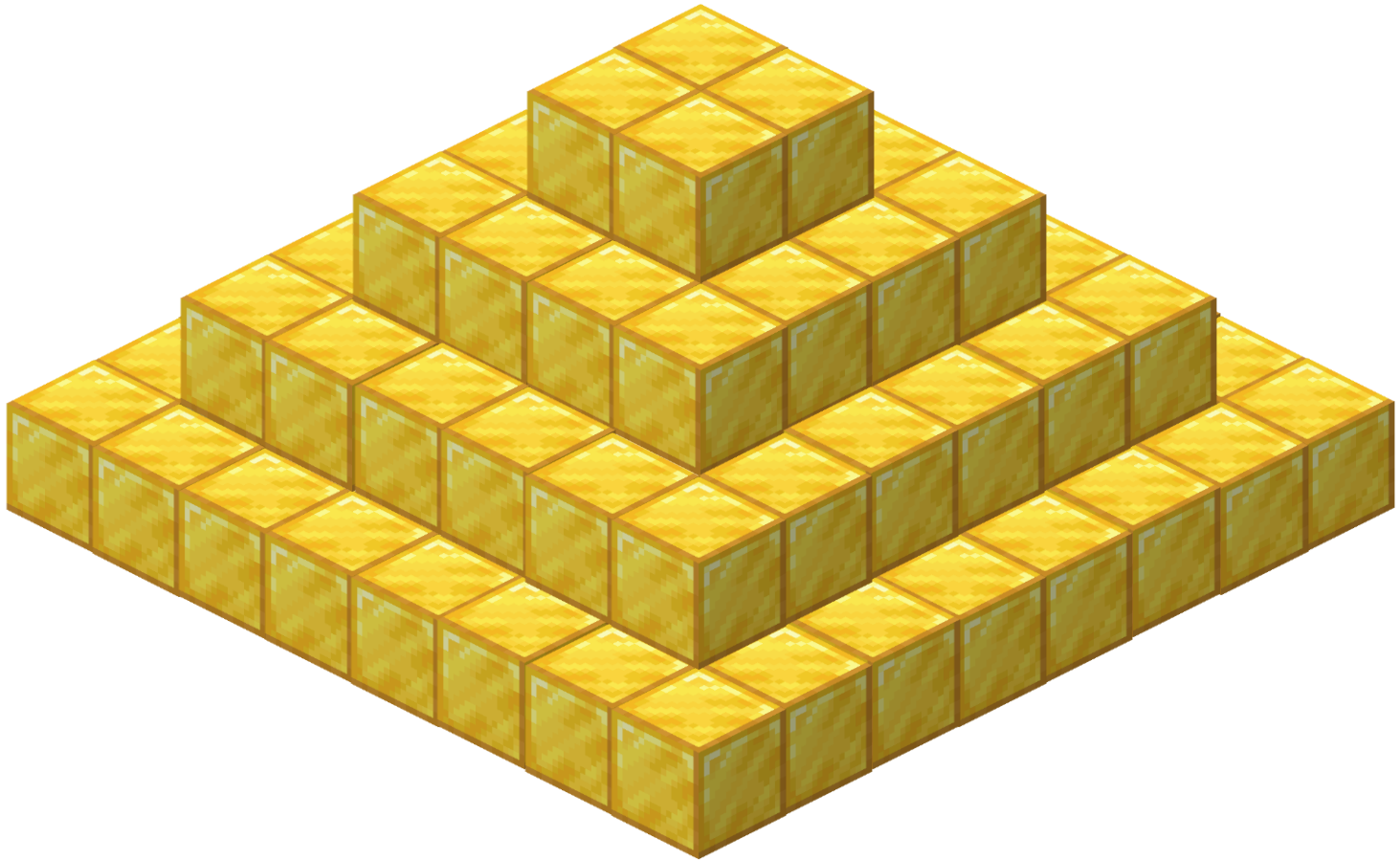


Cabin Structure



Cabin Roof

The Golden Pyramid



```
module "pyramid" {

  source = "./modules/pyramid"

  material = "minecraft:diamond_block"
  length   = 20

  start_position = {
    x = -1600,
    y = 62,
    z = -1101
  }

}
```

kcdc.qonq.io:25565

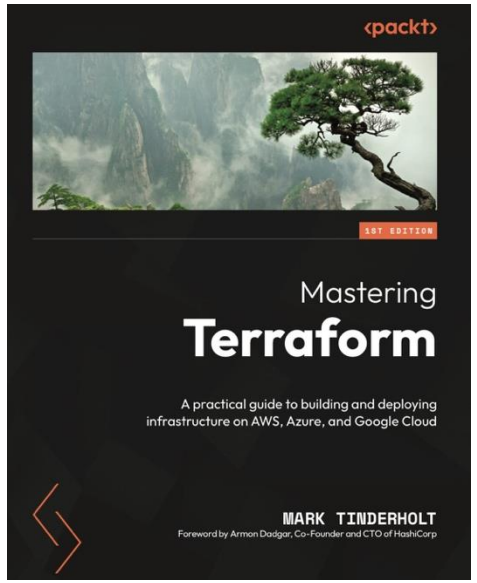
Key Takeaways

- **Have fun learning:** Minecraft + Terraform makes IaC tangible – build a template, see the world change.
- **IaC is for devs:** Not just a platform-engineer game. Start small, commit to Git, iterate.
- **Modularity FTW:** Terraform modules encapsulate proven patterns – clear inputs/outputs, composition, versioning, tests – so pieces click together (application environments or Minecraft builds).
- **Be the hero at home & work:** Host the family Minecraft server or ship reliable production environment – same skills, double win.


So, Stop Punching Trees!!!
...and Start Mining for Diamonds!!!

@MarkTinderholt

“Mastering Terraform”



Terraform 101: The Ultimate Hands-On Guide [Azure Edition]

A Practical, Step-by-Step Guide to Building and Automating Azure…

Mark Tinderholt

Premium Bestseller 4.8