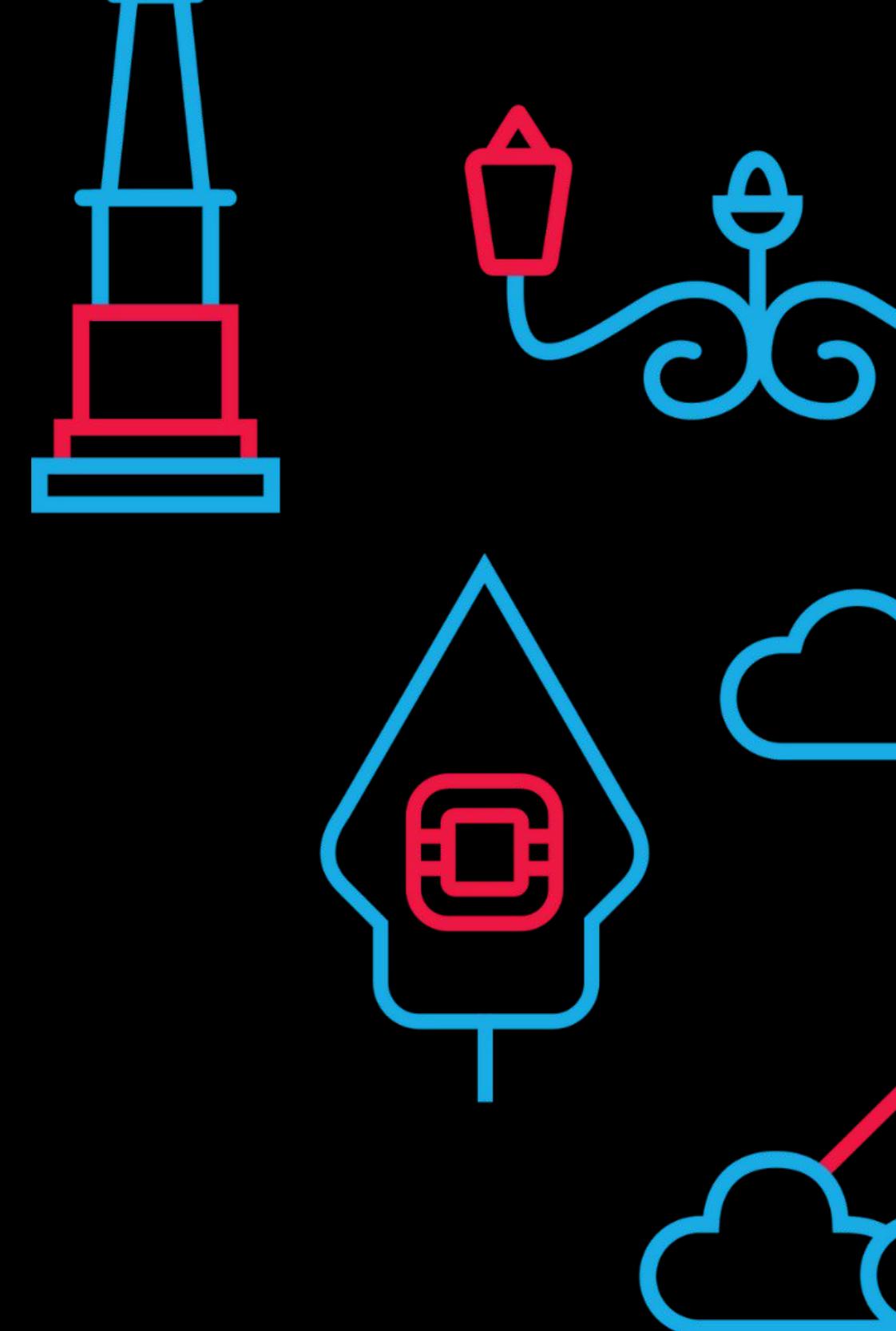


LESSON LEARNED: THINGS YOU SHOULD KNOW BEFORE YOUR FIRST IaC SETUP



NI PUTU SINTIA WATI

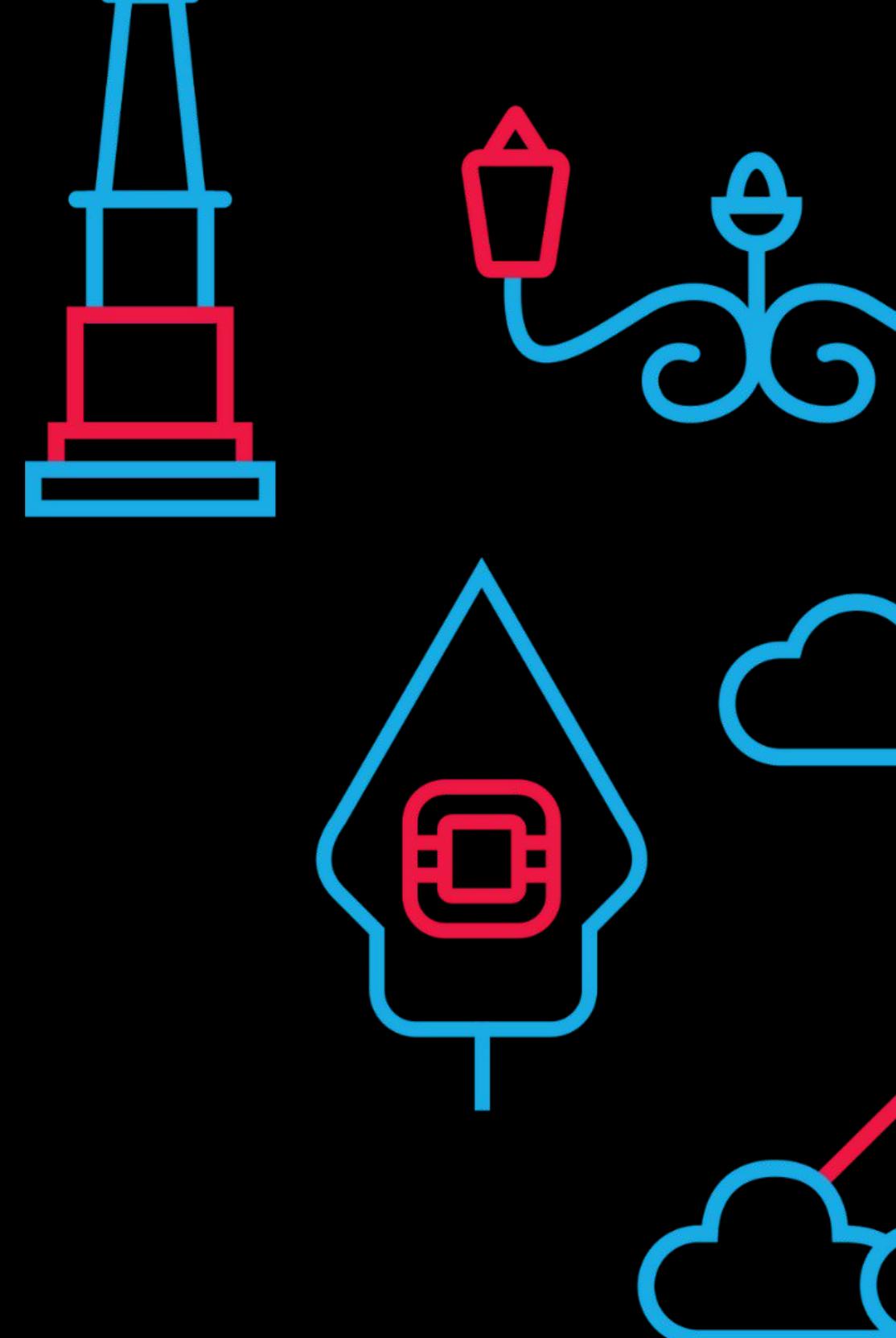
Infrastructure Engineer @Zero One Group



PEMBELAJARAN: HAL-HAL YANG PERLU KETAHUI SEBELUM SETUP IaC PERTAMAMU

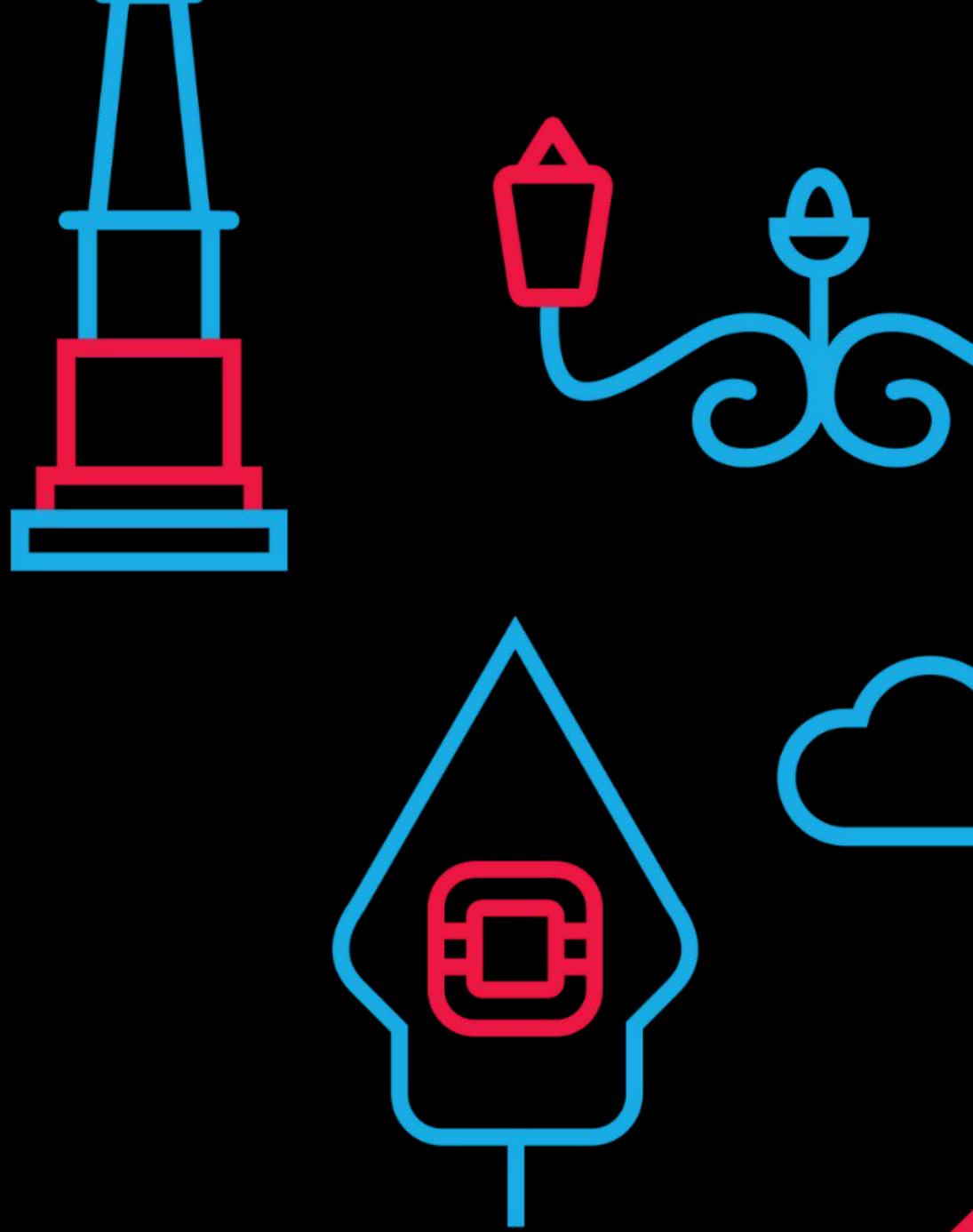
NI PUTU SINTIA WATI

Infrastructure Engineer @Zero One Group



DISCLAIMER

- *The views and lessons shared in this session are based on my personal experience.*
- *The tech stack featured today is OpenTofu (an open-source alternative to Terraform) running on DevStack as our on-premise cloud server.*





my resource today in: [**https://linktr.ee/SintialnOID2025**](https://linktr.ee/SintialnOID2025)



TODAY'S PLAN

- Intro
- Define
- Provision
- Manage

Intro What is Infrastructure as Code?



What is Infrastructure as Code?

IaC is the way to **define, provision, and manage** your infrastructure by writing code using a configuration tools, rather than doing it manually through a web interface or running commands via SSH.

Why Infrastructure as Code?

Declarative

Instead of specifying the exact steps to make a change, you declare the desired end state of the infrastructure, and the IaC tool handles the implementation details. This allows anyone to understand the state of the infrastructure by reading the code.



Code Management

IaC is managed like application source code, allowing for version control (commit, version, track, collaborate). Version control also facilitates collaboration among developers.

Auditable

IaC provides an auditable history of your infrastructure, allowing you to include explanations (comments) for changes. This gives a clear view of how the infrastructure has evolved and allows for previewing new changes to detect drift before deployment.



Portable

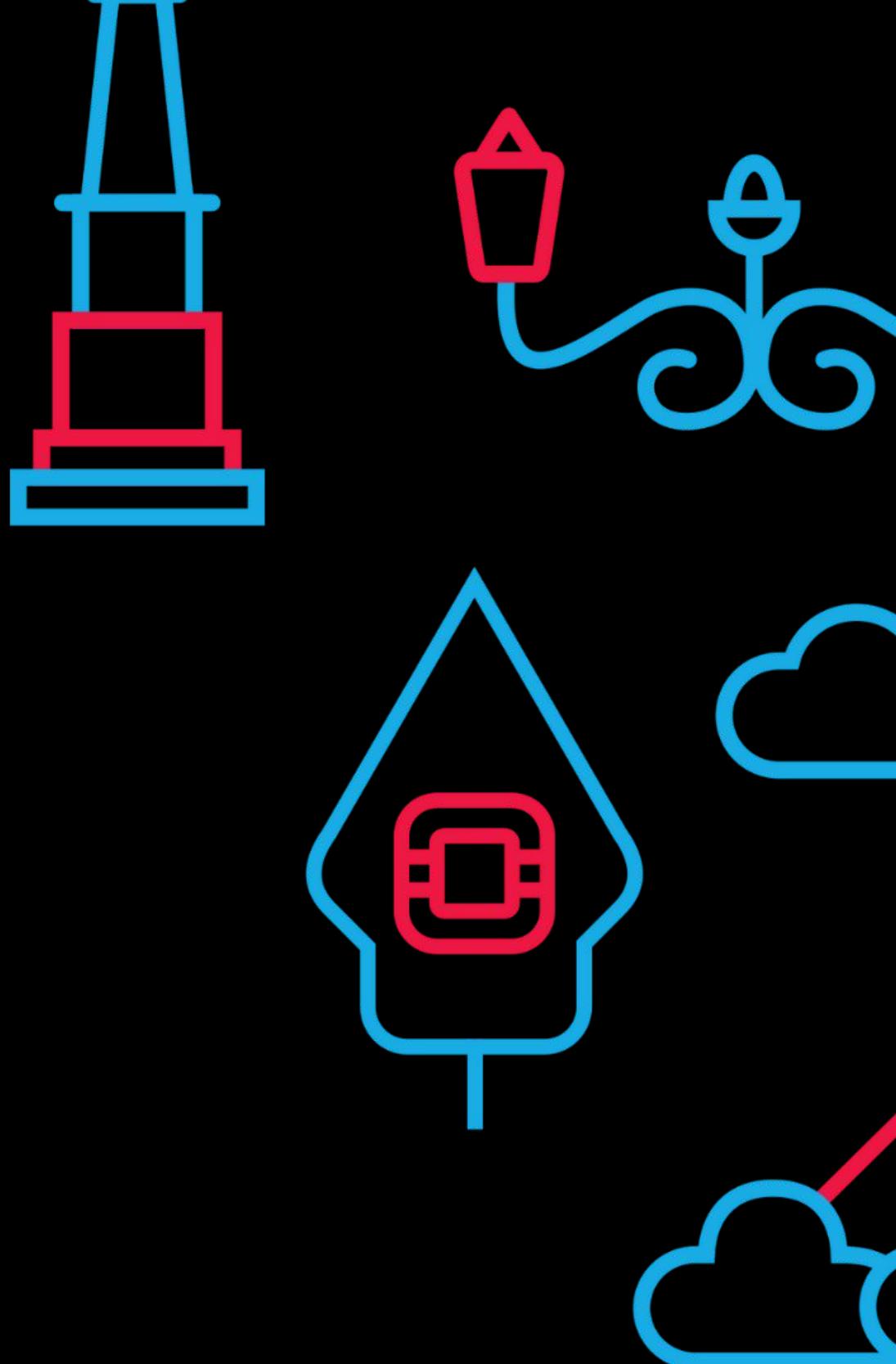
You can build reusable modules that encapsulate conventions, enabling consistent infrastructure building from a shared template. This allows for deploying many instances of the same template for different applications or regions without manual rebuilding.



Common configuration tools

- Terraform (alternative is OpenTofu)
- Ansible
- Chef
- Puppet
- Pulumi

Fase 1: Define



*This is where you **write code** to declare the desired state of your infrastructure.*

- **What you do:** Author human-readable configuration files
- **What you specify:** The resources you need (servers, networks, databases) and their exact configurations (size, region, version).
- **The Goal:** To create a "blueprint" that acts as the single source of truth for your infrastructure. At this stage, nothing is built yet.

11

,”



```
resource "openstack_objectstorage_container_v1" "states_bucket" {  
  name = "openstack-state-buckets"  
  metadata = {  
    "purpose" = "terraform-state"  
  }  
}
```

Lesson: Failing to plan is planning to fail.

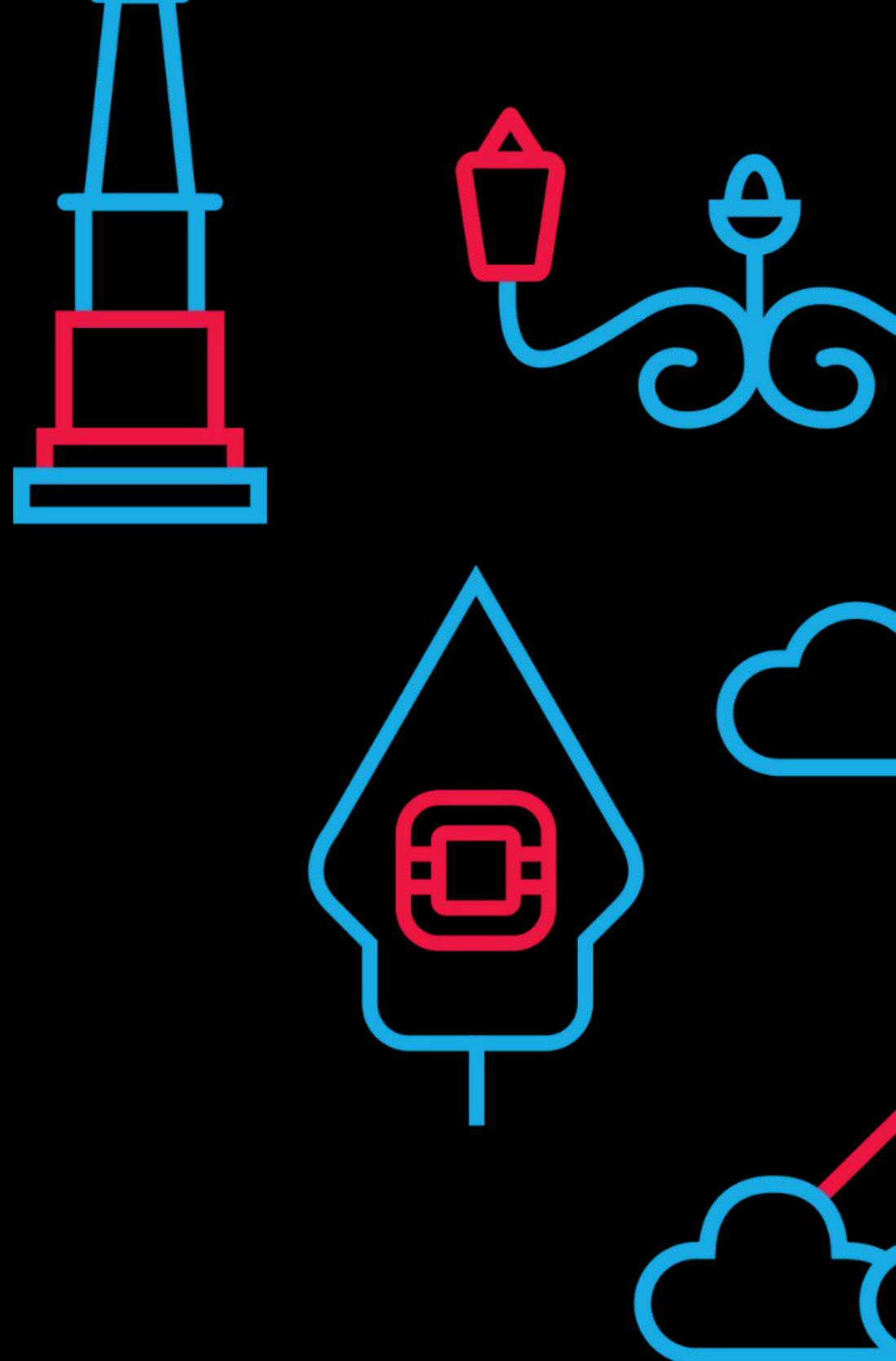
Understand the Documentation: It's your best friend. Before writing any code, understand the resources and arguments provided by OpenTofu and your provider.

Understand the Service: Know what you want to build (e.g., Compute, Network, DB) before deciding how to build it. Different services require different configurations.

Different Provider, Different 'Language': Understand that each provider has unique authentication methods and resource naming ¹³conventions. Prepare the correct and secure credentials from the start.

“ ”

Fase 2: Provision

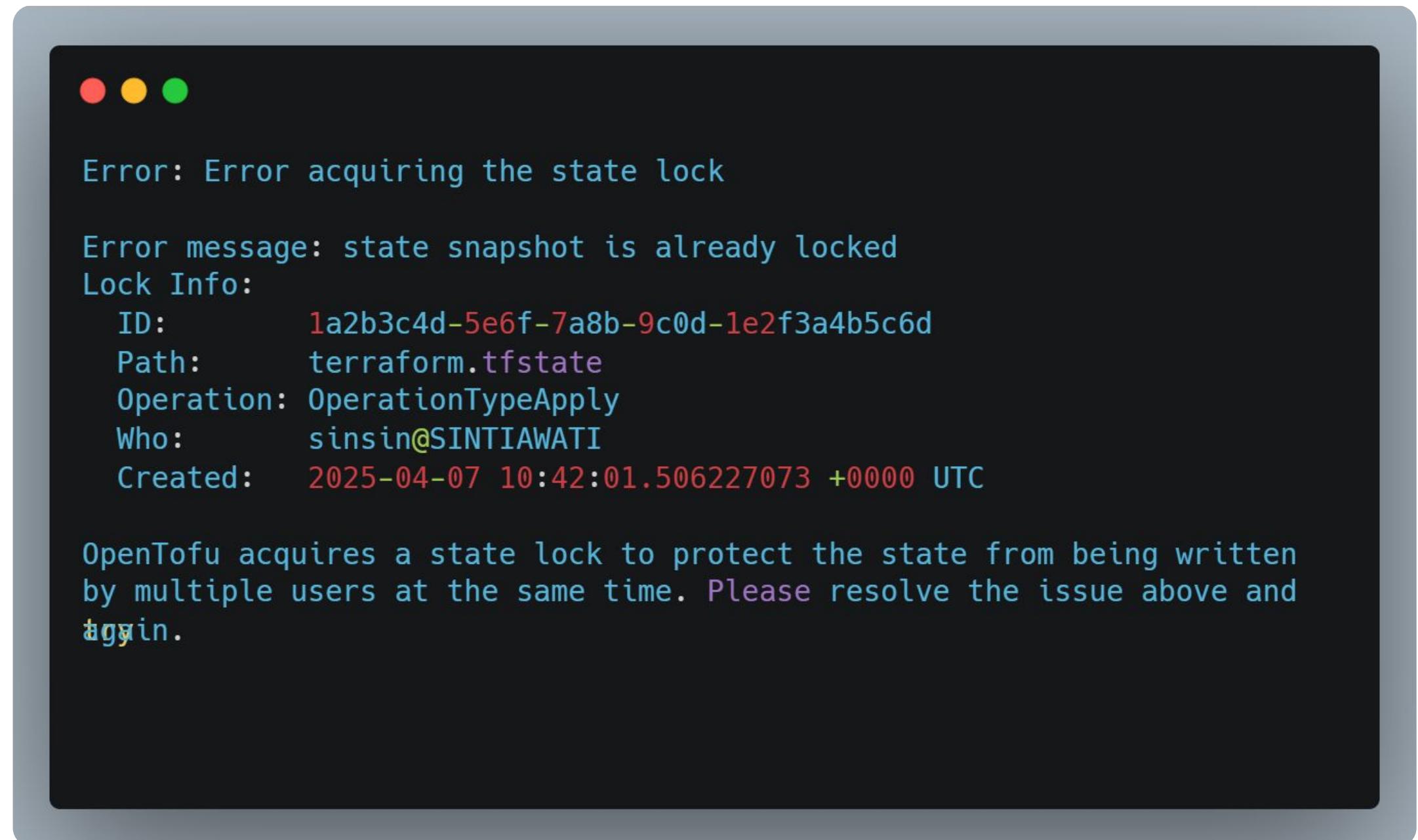


This is the action phase where the **IaC tool builds your infrastructure** based on the blueprint.

- **What you do:** Run a command like `tofu apply` or `terraform apply`.
- **What the tool does:** It reads your code, communicates with your cloud provider's API, and creates the real, running resources.
- **The Goal:** To transform your code into a tangible, operational environment automatically and reliably.

15

,”



Lesson: "Trust, but verify." Never apply blindly.

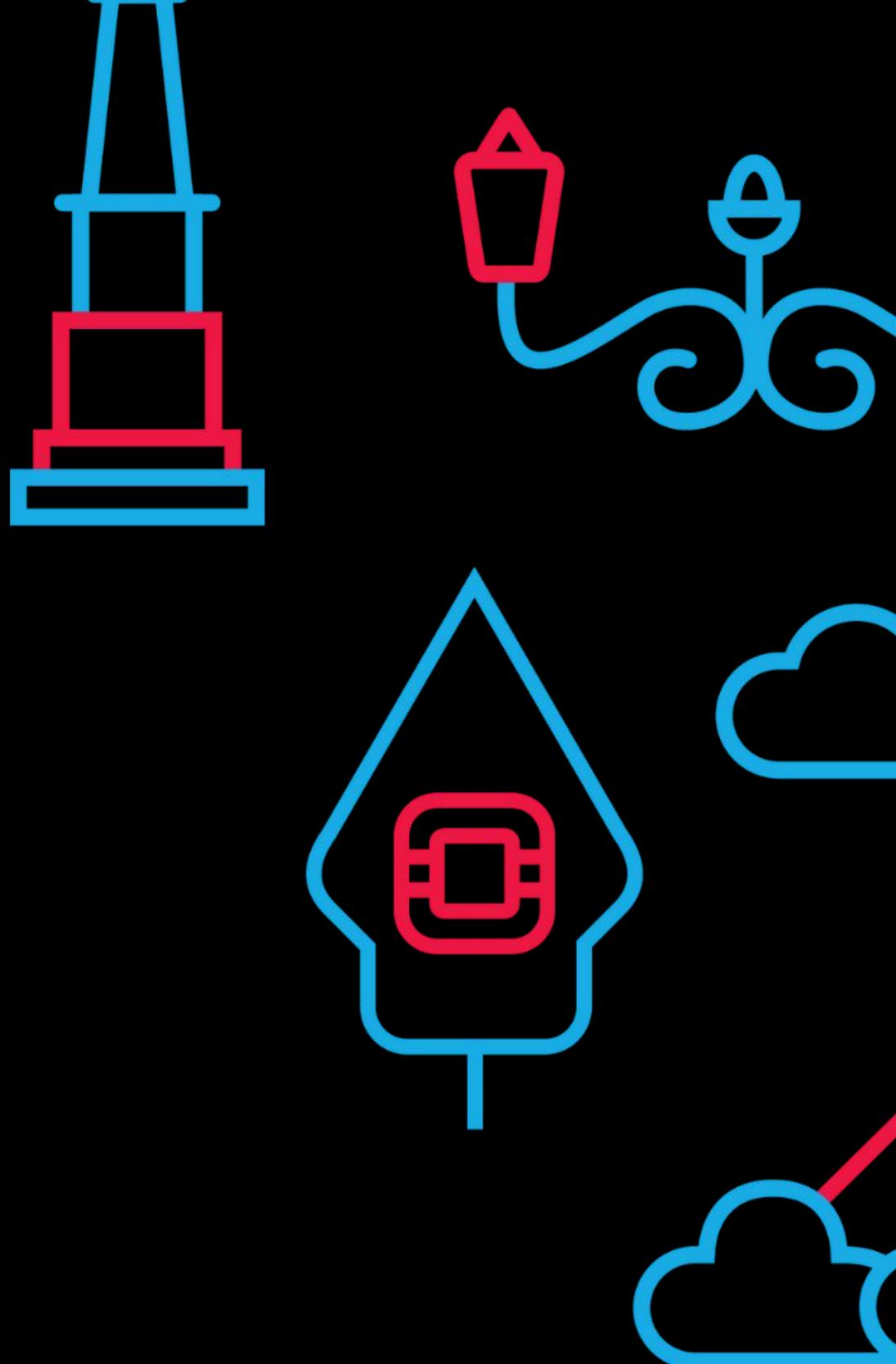
plan is Your Safety Net: Always run *tofu plan* and analyze its output carefully. Understand what will be created, changed, or destroyed.

Don't apply in the Dark: Make sure you know the exact impact of the *tofu apply* command. In production, a small mistake can have a huge impact.

17

”

Fase 3: Manage



This phase covers everything that happens after the *initial creation*. Infrastructure is never static.

- **What you do:** Modify your code to update, scale, or change the infrastructure.
- **What the tool does:** It intelligently applies only the necessary changes, tracks the state of your resources, and can cleanly destroy everything when it's no longer needed.
- **The Goal:** To handle the entire lifecycle—from day-one updates to final decommissioning—through code, ensuring consistency and control over time.



Lesson: Infrastructure is a living thing. Prepare for change and collaboration.

Use Remote State From Day One: If there's any chance of a handover or teamwork, don't delay. Move your state from local to a remote backend (like S3) to avoid conflicts and state loss.

Use -target Wisely: This feature is very useful for emergencies, but it's dangerous if it becomes a habit. Use it only when you are absolutely sure.

Write Code for Humans: Good code isn't just executable by machines; it's also easily understood by your colleagues during a handover. Use clear names and a clean structure.

“ ”

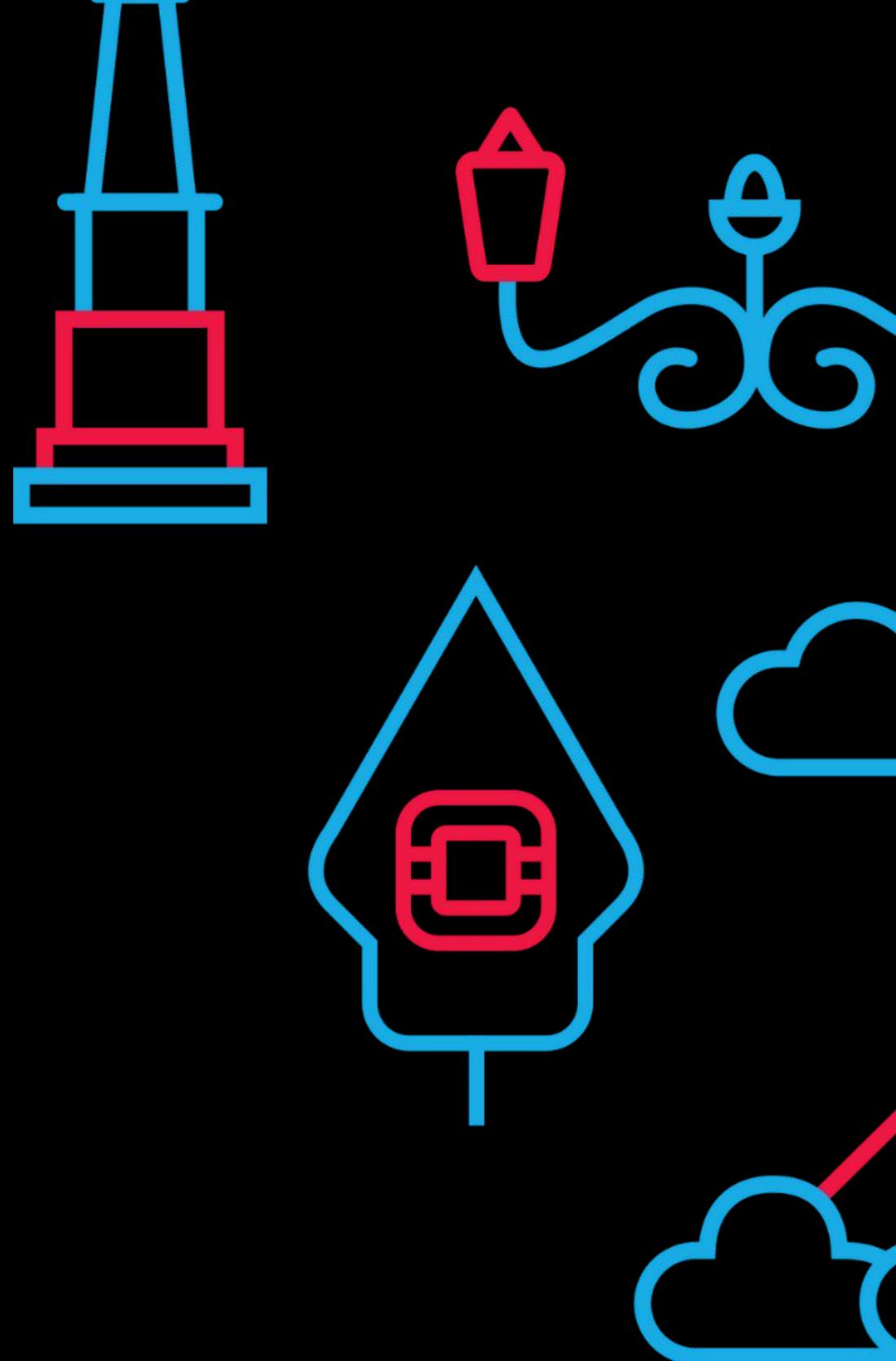
Key Takeaways

DEFINE: *Plan thoroughly. Invest time in reading documentation and understanding the architecture before writing code.*

PROVISION: *Make tofu plan a mandatory ritual. Confidence comes from verification, not assumption.*

MANAGE: *Think long-term. Use remote state²¹ for collaboration and write clean code for an easy handover.*

THANK YOU



Yogyakarta, 19 July 2025