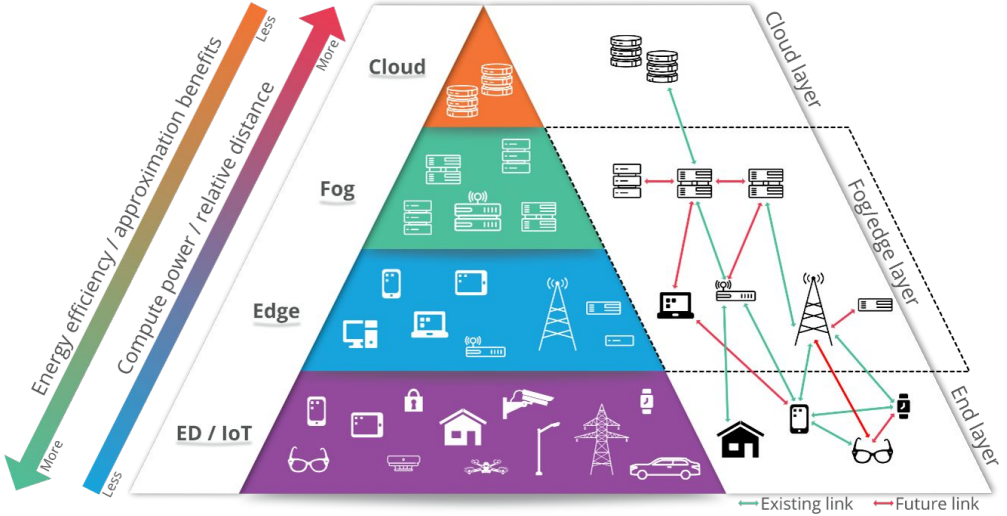
DISTRIBUTED AND CLOUD COMPUTING

LAB 14: CLOUD BASICS + IAC

(Module: K8S & CLOUD BASICS)

Cloud Computing

- "A paradigm for enabling network access to a scalable and elastic pool of shareable physical or virtual resources with self-service provisioning and administration on-demand."
- 5 Essential Characteristics (by INST):
  a. **On-demand self-service**
     - Provision as needed
     - Automatic provisioning
  b. **Broad network access**
     - Heterogeneous clients
  c. **Resource pooling**
     - Serve multiple clients
  d. **Rapid elasticity**
     - "Unlimited" resources
  e. **Measured service**
     - Monitoring for optimization
- Future: End-Edge-Cloud Hierarchy



https://en.wikipedia.org/wiki/Cloud_computing
https://projects.tuni.fi/apropos/esr-papers/damsgaard2022approximation/

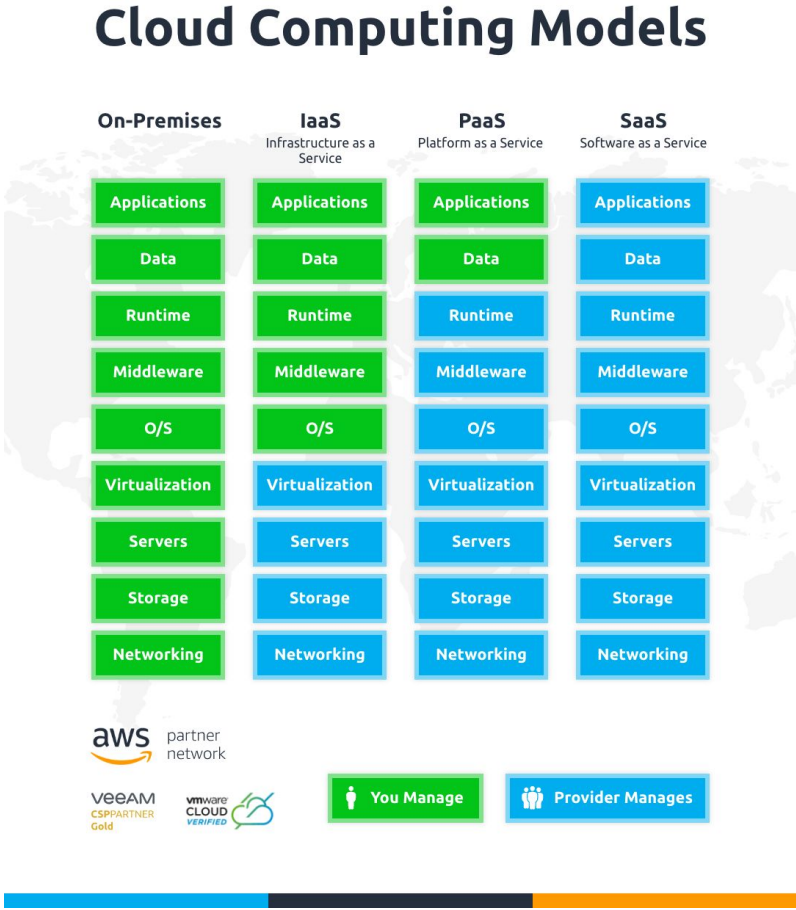
Cloud Providers

- “A CSP (cloud service provider) is a third-party company that provides scalable computing resources that businesses can access on demand over a network, including cloud-based compute, storage, platform, and application services.”
- Common Providers:
  - Amazon Web Services (AWS)
  - Google Cloud Platform (GCP)
  - Microsoft Azure (AZR)
  - In China:
    - Alibaba Cloud
    - Tencent Cloud
    - Huawei Cloud
  - …

https://cloud.google.com/learn/what-is-a-cloud-service-provider
https://dev.to/dkechag/cloud-vm-performance-value-comparison-2023-perl-more-1kpp

Cloud Service Models

- **Infrastructure as a Service (IaaS)**
  - “on-demand access to cloud-hosted physical and virtual servers, storage and networking”
- **Platform as a Service (PaaS)**
  - “on-demand access to a complete, ready-to-use, cloud-hosted platform for developing, running, maintaining and managing applications”
- **Software as a Service (SaaS)**
  - “on-demand access to ready-to-use, cloud-hosted application software”
- … (FaaS, CaaS, KaaS, etc.)

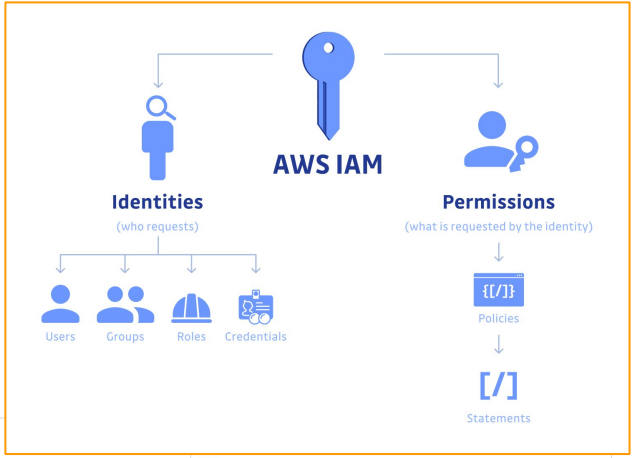


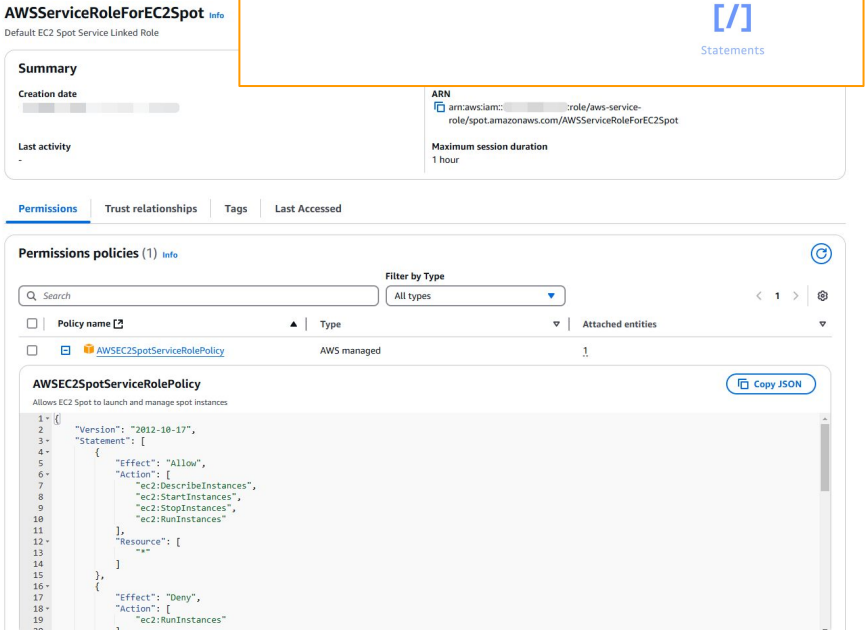
https://www.ibm.com/think/topics/iaas-paas-saas
https://www.heptabit.at/blog/cloud-migration/pros-and-cons-of-different-cloud-computing-models

Common Cloud Services

Identity & Access Management (IAM)

- "The user has a responsibility and is allowed to perform these operations on these resources."
- General Concept
  - **Principal/Identity**
    - User accounts
    - Service accounts
    - OIDC identity providers
    - …
  - **Role**
    - collection of permissions
- Example
  - AWS IAM
  - Google Cloud IAM
  - Microsoft Entra ID



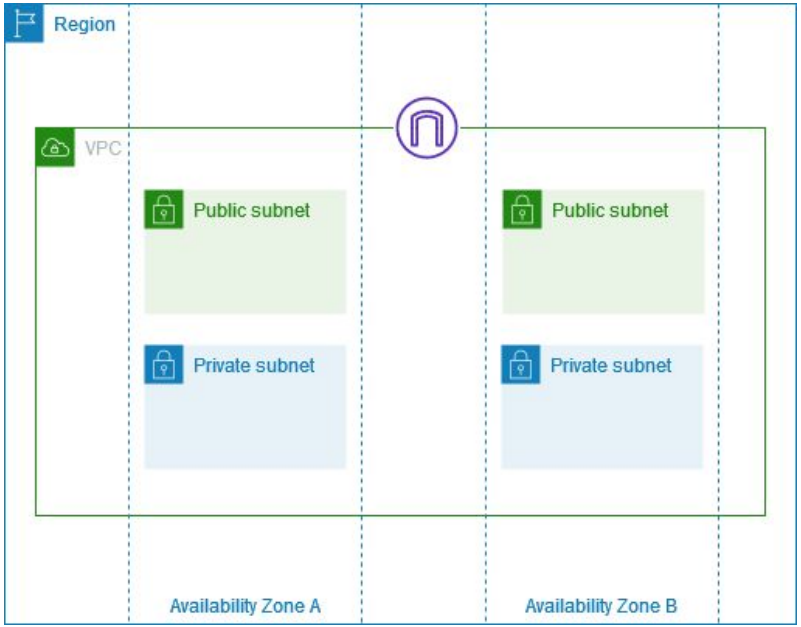


https://cloud.google.com/iam/docs/overview#how_cloud_iam_works
https://k21academy.com/amazon-web-services/aws-identity-and-access-management-iam/

Common Cloud Services

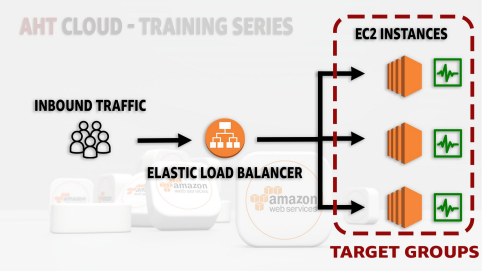
Networking

- Pre-requisites (configured by cloud providers)
  - **Region**: independent geographic areas that consist of zones.
  - **Availability Zone (AZ)**: deployment area housed in physical data centers.
- **Virtual Private Cloud (VPC)**
  - A logically isolated virtual network.
  - **Subnet**: a range of IP addresses in a VPC.
    - **Public Subnet**: possesses a route to an internet gateway in a route table.
    - **Private Subnet**
  - Example
    - AWS VPC
    - Google VPC
    - Azure Virtual Network



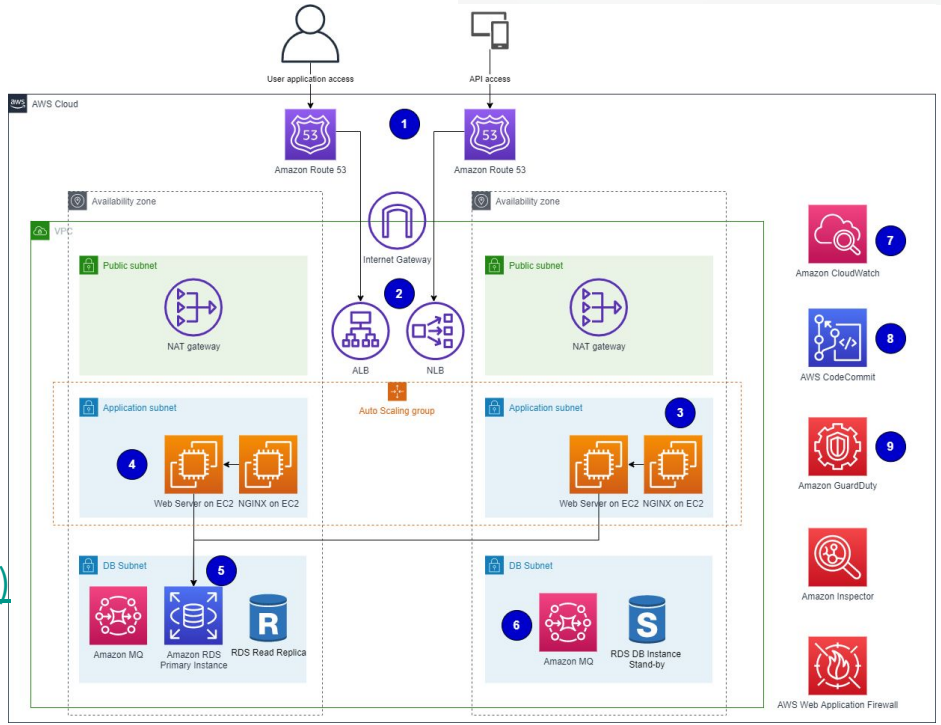
https://cloud.google.com/compute/docs/regions-zones
https://docs.aws.amazon.com/vpc/latest/userguide/configure-subnets.html
https://docs.aws.amazon.com/vpc/latest/userguide/how-it-works.html#how-it-works-subnet

Common Cloud Services



Networking

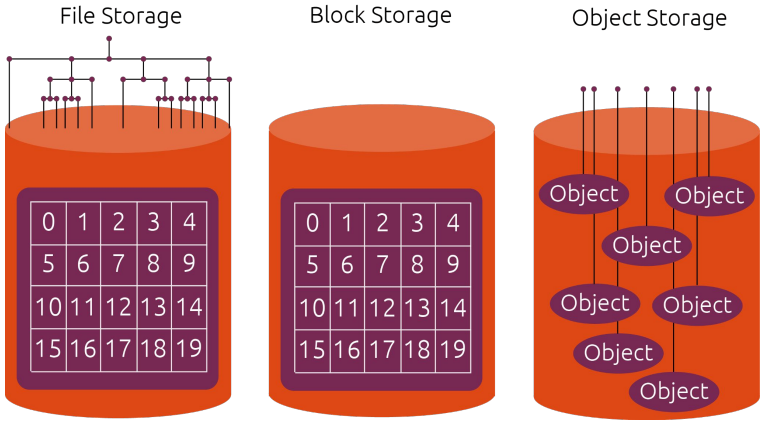
- Pre-requisites
  - **Region**
  - **Availability Zone (AZ)**
- **Virtual Private Cloud (VPC)**
- **DNS**
  - Example
    - AWS Route 53
    - Google Cloud DNS
    - Azure DNS
- **Load Balancing**
  - Example
    - AWS ELB (Elastic Load Balancing)
    - Google Cloud Load Balancing
    - Azure Load Balancer



https://www.ahtcloud.com/what-is-elastic-load-balancing

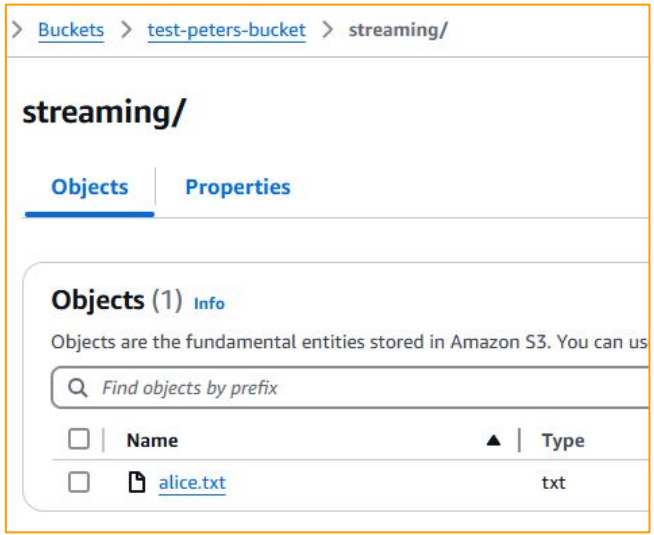
https://aws.amazon.com/blogs/architecture/how-munich-re-automation-solutions-ltd-built-a-digital-insurance-platform-on-aws/

Common Cloud Services



Object Storage

- "Object Storage stores and manages data in an unstructured format called **objects**."
- High throughput for large & durable data transfer.
- Example
  - AWS S3 (Simple Storage Service)
  - GCS (Google Cloud Storage)
  - Azure Blob Storage



Block Storage

- "Block Storage stores data **blocks** in a manner that is optimized for fast access and retrieval."
- High IOPS & low latency for transactions.
- Example
  - AWS EBS (Elastic Block Store)
  - Google Persistent Disk
  - Azure Managed Disks

https://canonical.com/blog/what-are-the-different-types-of-storage-block-object-and-file

#GCPSketchnote
@PVERGADIA
THECLOUDGIRL.DEV
04.23.2021

Which Storage Should I Use?

OBJECT

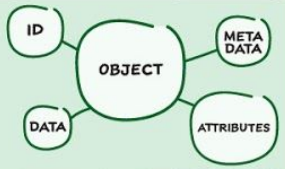
Cloud Storage

For any app. Store any type & any amount of data for any duration, & retrieve it as often as needed

GOOD FOR:
Binary or object data, blobs, unstructured data

Object Storage

DATA ↓ ↑ METADATA/ OBJECT ID



USE CASE:

- Streaming videos
- Images
- Data analytics
- Backups
- Documents
- Regulatory archives
- Tape replacement
- Websites
- Genomics
- Disaster recovery

BLOCK

Persistent Disk ⟷ Fully integrated with Compute Engine & GKE ⟷ **Local SSD**

Block Storage

BLOCK# ↓ # ↑ BLOCK#

GOOD FOR:
- ✓Block store for VMs
- ✓Range of latency & performance options

USE CASE:

- Disks for VMs
- Share read-only data across VMs
- Rapid, durable backups of running VMs
- Storage for databases

GOOD FOR:
- ✓Ephemeral block store for VMs
- ✓Lowest latency
- ✓Stateless workloads

USE CASE:

- Flash-optimized databases
- Hot caching layer for analytics
- Application scratch disk
- Scale out analytics
- Media rendering

FILESTORE

Filestore

Fully managed, cloud-based Network Attached Storage

GOOD FOR:
Shared file storage (unstructured) data

USE CASE:

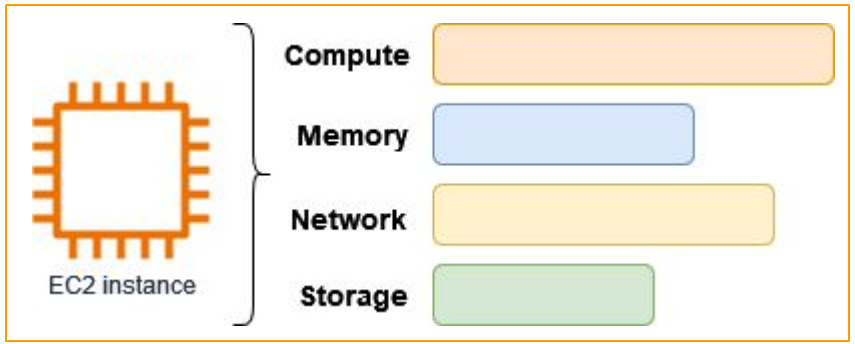
- Media processing
- Life sciences/ Genomics processing
- Electronic Design Automation (EDA)
- Data Analytics
- Application migrations
- Machine Learning
- Web content management
- Financial Modeling

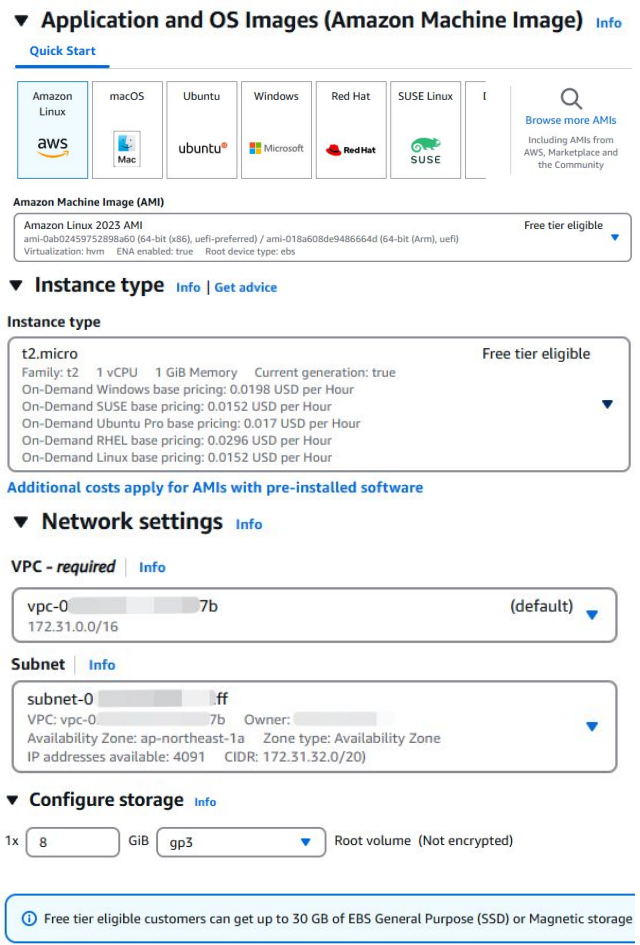
https://cloud.google.com/blog/topics/developers-practitioners/map-storage-options-google-cloud

Common Cloud Services

Compute

- A virtual machine (VM) as a compute instance.
- Uses Block Storage as “Volume”.
- Use VPC for networking.
- Example
  - AWS EC2 (Elastic Compute Cloud)
  - GCE (Google Compute Engine)
  - Azure VMs





https://cloud.google.com/compute/docs/instances
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

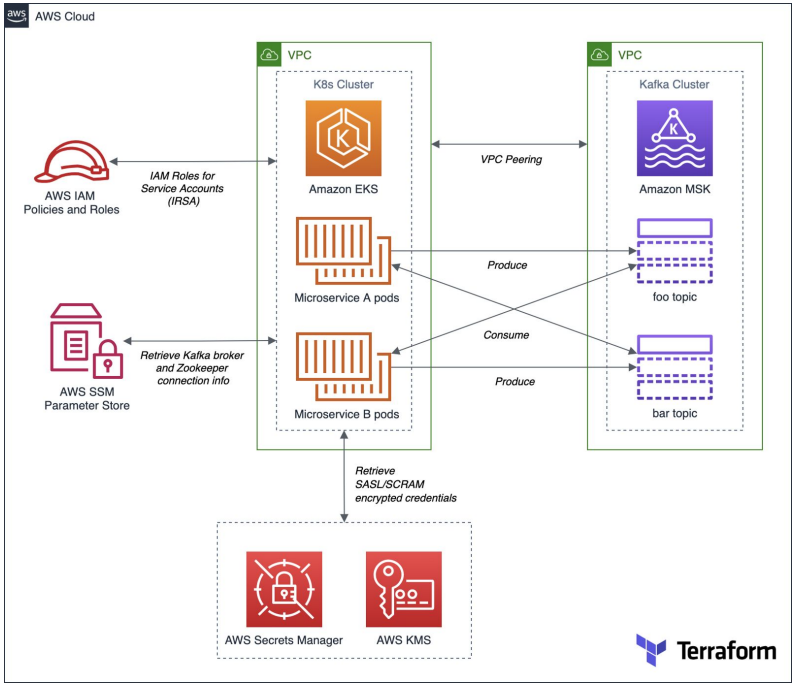
Common Cloud Services

Container Orchestration

- **Kubernetes as a Service (KaaS)**
  - Implements the Cloud Controller Manager component from Kubernetes control plane.
    - E.g., interact with EC2, ELB, EBS.
  - Example
    - AWS EKS (Elastic Kubernetes Service)
    - GKE (Google Kubernetes Engine)
    - AKS (Azure Kubernetes Service)

Serverless Compute

- **Function as a Service (FaaS)**
  - Build & run apps without managing servers.
  - Example
    - AWS Lambda
    - Google Cloud Run Functions
    - Azure Functions



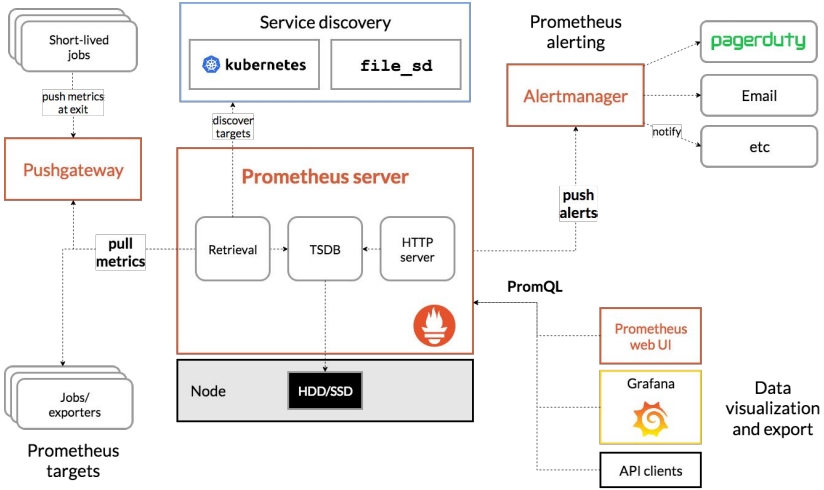
https://programmaticponderings.com/2021/07/26/securely-decoupling-applications-on-amazon-eks-using-kafka-with-sasl-scram/

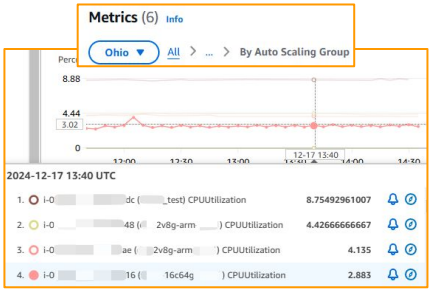
We will stop here for common cloud services. There are amazingly so much more to explore if you are interested!

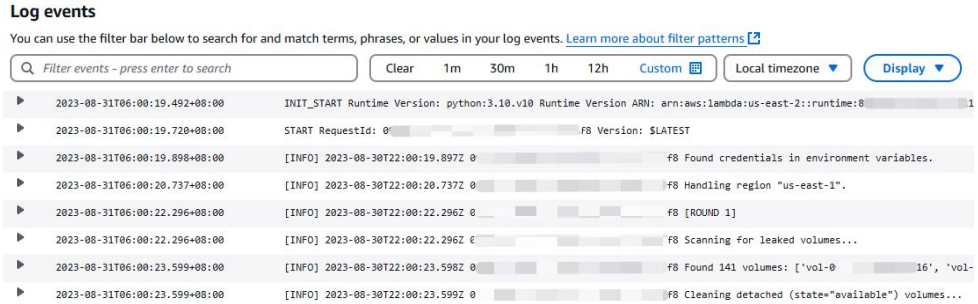
Common Cloud Services

Monitoring & Logging

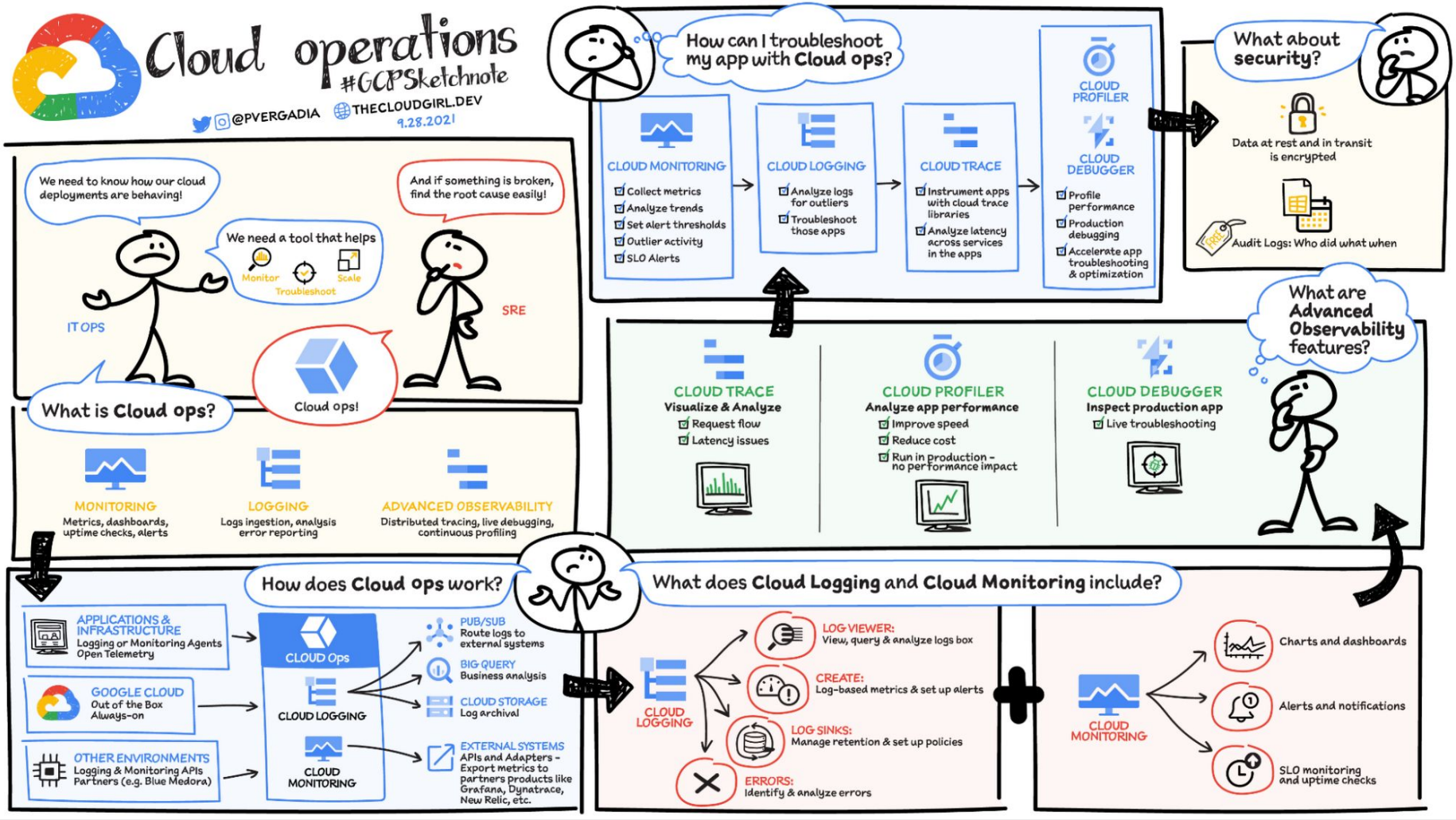
- Remember the Measured Service characteristic of Cloud Computing?
- Monitor resources for automated optimizations.
- For Kubernetes, we have Prometheus.
  - In AWS → AMP (Managed Prometheus)
- Cloud providers also have their own solutions.
  - Example
    - AWS CloudWatch
    - Google Cloud's Observability
    - Azure Monitor







https://prometheus.io/docs/introduction/overview/#architecture



https://cloud.google.com/blog/topics/developers-practitioners/introduction-google-clouds-operations-suite

Infrastructure as Code

- "**Infrastructure as Code (IaC)** is the managing and provisioning of infrastructure through code instead of through manual processes."
- **Declarative (preferred)** vs. Imperative IaC
- Benefits
  a. Reduced cost, time, and risk of deployments.
  b. Eliminates configuration drift to ensure infrastructure consistency.
- Common IaC Tools
  a. Helm: manages **K8s apps** using **Charts** as config/template
  b. Terraform: for now one of the most popular IaC tools
  c. OpenTofu: an **open-source** fork of Terraform
  d. Cloud Solutions
     - AWS CloudFormation
     - Google Cloud Deployment Manager
     - Azure Resource Manager
  e. …

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: mychart-configmap
data:
  myvalue: "Hello World"
```

HELM

HashiCorp Terraform

OpenTofu

Helm Chart

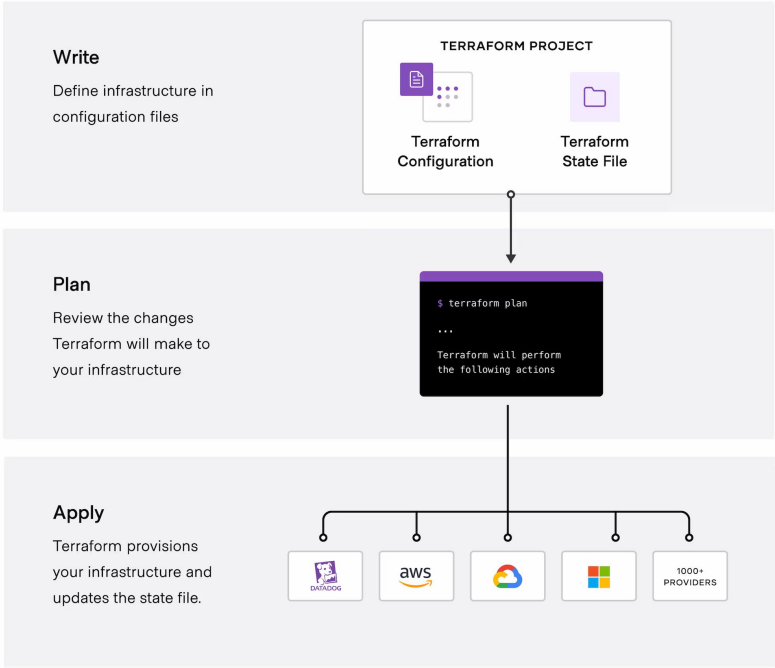
```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: {{ .Release.Name }}-configmap
data:
  myvalue: "Hello World"
```

https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac

Terraform as an IaC Tool

- Developed by HashiCorp.
- Workflow Stages: Write → Plan → Apply
- Features:

1. Manages any infrastructure
   a. Multi-cloud / hybrid cloud / on-premise
   b. Provider: plugins to interact with cloud providers, SaaS providers, etc.
   c. Terraform Registry: central repository for provider & module sharing.
2. Drift Detection & Management
   a. Drift: difference between the real-world infrastructure state & the state defined in the configuration files.
   b. State: stores the current infrastructure state in a state file.

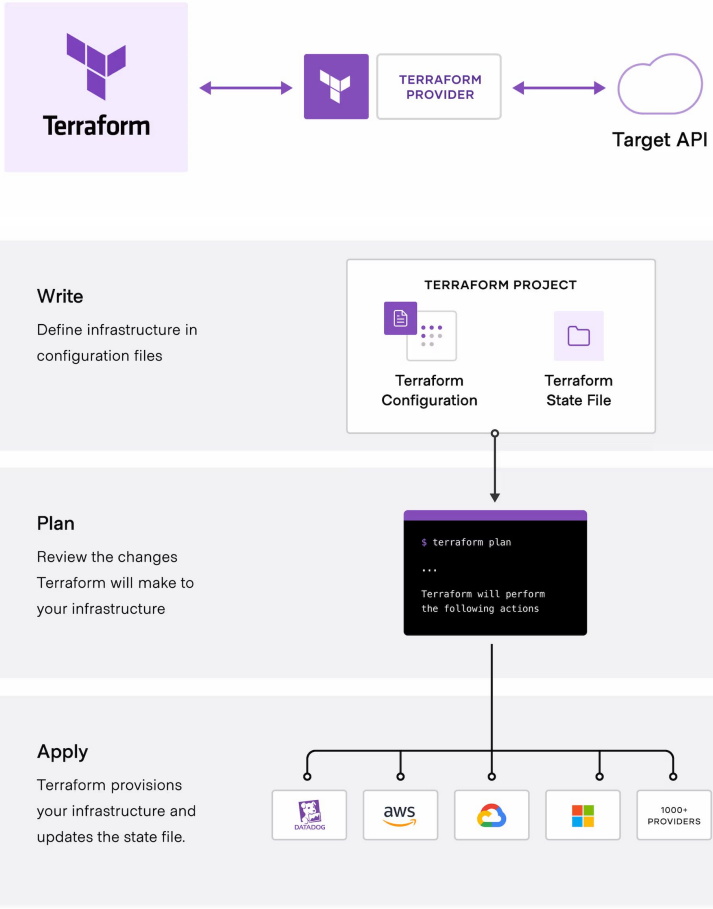


https://developer.hashicorp.com/terraform/intro

Terraform as an IaC Tool

- Developed by HashiCorp.
- Workflow Stages: Write → Plan → Apply
- Features:

1. Manages any infrastructure
2. Drift detection & management
3. Automated efficient changes
   a. Declarative `.tf` configuration files using the Terraform Language.
   b. Resource graph for resource dependency determination.
   c. Handles independent resources in parallel.
4. Standardizes configurations
   a. Declarative `.tf` configuration files using the Terraform Language.
   b. Modules as reusable components.



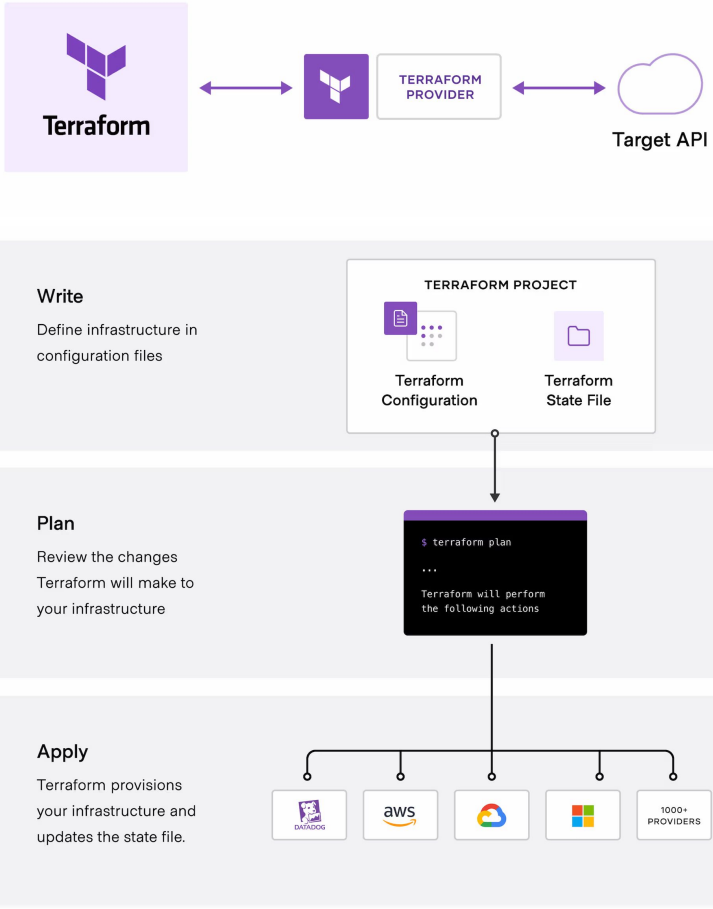
https://developer.hashicorp.com/terraform/intro

Terraform as an IaC Tool

- Developed by HashiCorp.
- Workflow Stages: Write → Plan → Apply
- Features:

1. Manages any infrastructure
2. Drift detection & management
3. Automated efficient changes
4. Standardizes configurations
5. Supports VCS & Collaboration
    a. IaC: infrastructure config as code files
    b. State - states are also stored in state files
    c. State Locking - locks the state file to prevent concurrent writes

We currently have no money to play with the cloud providers, but we can try a local file demo to at least understand how Terraform works...



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

TASK: Terraform

Note how the states are maintained and how state locking is performed.

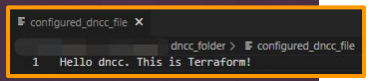
Use Terraform to manage a local file.

> Reference codebase: `iac_tf_local`

1. Install Terraform CLI.
2. Check the `.tf` configuration files in the `dncc_file/` folder.
   - This folder is considered as a Terraform module.
   - All these `.tf` files (names do not matter) will be executed by Terraform.
   - It contains the provider specification, the local file configuration, as well as the input & output formats.
   - The `terraform.tfvars` file sets the values of the input variables specified in the configurations.
3. Initialize the Terraform module: `terraform init`
4. Apply the module and confirm changes: `terraform apply`
5. The file will be created at the expected path. Now manually modify the file and apply again to check drift detection.
6. Clean up the resources after testing: `terraform destroy`
7. (*) Try the official tutorials (e.g., AWS) if available.

```
Initializing the backend...
Initializing provider plugins...
- Finding hashicorp/local versions matching "2.5.2"...
- Installing hashicorp/local v2.5.2...
- Installed hashicorp/local v2.5.2 (signed by HashiCorp)
Terraform has created a lock file .terraform.lock.hcl to record the provider
selections it made above. Include this file in your version control repository
so that Terraform can guarantee to make the same selections by default when
you run "terraform init" in the future.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

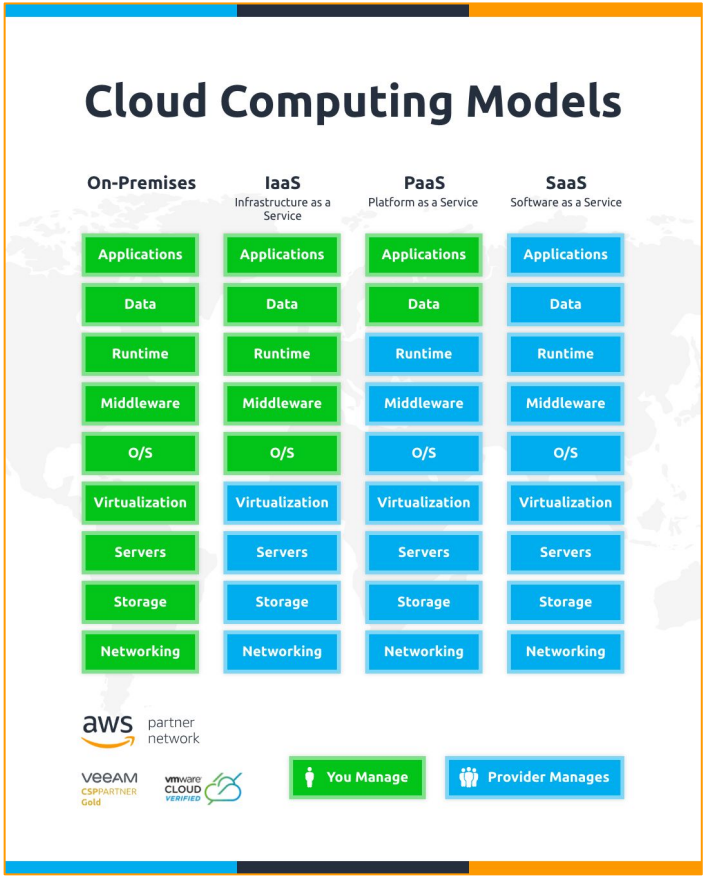
```
cd dncc_file/; terraform apply

Terraform used the selected providers to generate the following execution plan. Resource acti
  + create

Terraform will perform the following actions:

  # local_file.my_file will be created
  + resource "local_file" "my_file" {
      + content              = "Hello dncc. This is Terraform!"
      + content_base64sha256 = (known after apply)
      + content_base64sha512 = (known after apply)
      + content_md5          = (known after apply)
      + content_sha1         = (known after apply)
      + content_sha256       = (known after apply)
      + content_sha512       = (known after apply)
      + directory_permission = "0777"
      + file_permission      = "0777"
      + filename             = "../dncc_folder/configured_dncc_file"
      + id                   = (known after apply)
    }

Plan: 1 to add, 0 to change, 0 to destroy.

Changes to Outputs:
  + path = "../dncc_folder/configured_dncc_file"

Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

local_file.my_file: Creating...
local_file.my_file: Creation complete after 0s [id=d5057c22e81212273b0e425892ca84c33487f046]

Apply complete! Resources: 1 added, 0 changed, 0 destroyed.

Outputs:

path = "../dncc_folder/configured_dncc_file"
```



Summary

- **5 Essential Cloud Computing Characteristics**
  a. On-demand self-service
  b. Broad network access
  c. Resource pooling
  d. Rapid elasticity
  e. Measured service
- **Common Cloud Providers**: AWS, GCP, AZR, …
- **Cloud Service Model**: IaaS *vs.* PaaS *vs.* SaaS
- **Common Cloud Services**:
  a. IAM
  b. Networking: VPC, DNS, Load Balancing
  c. Object Storage *vs.* Block Storage
  d. Compute & Serverless Compute
  e. Container Orchestration
  f. Monitoring & Logging
- **IaC**: Helm, Terraform, OpenTofu, …

That's all for the K8s & Cloud Basics lab!



https://www.heptabit.at/blog/cloud-migration/pros-and-cons-of-different-cloud-computing-models