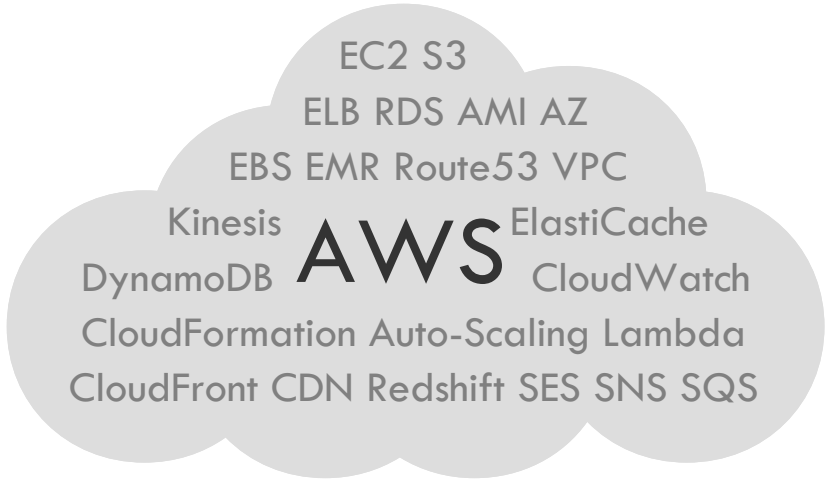
ARCHITECTING WEB APPLICATIONS FOR THE CLOUD: DESIGN PRINCIPLES AND PRACTICAL GUIDANCE FOR AWS

Dr Adnene Guabtni, Senior Research Scientist, NICTA/Data61, CSIRO
Adnene.Guabtni@csiro.au



LET'S CLEAR ANY MISUNDERSTANDING ABOUT THE CLOUD

29% of the general public think Cloud Technology is an actual cloud (Wakefield Research)

But hey, that's the general public! What about IT professionals?

Many IT professionals would think that the benefits of the cloud are:

- Portable office.
- Cost Savings.
- Fewer responsibilities, easier manageability.
- Reliability (SLA which guarantees 24/7/365 and 99.99% availability).

Looks like traditional "Hosted services".

Cloud Computing is Different.

THE CLOUD COMPUTING DIFFERENCE

❶ IT Assets Become Programmable Resources

- Servers, databases, storage, and higher-level application components are temporary and disposable, quickly provisioned when needed.
- They dynamically scale to meet actual demand.
- You only pay for what you use.

Think of how a software allocates memory on demand and "garbage collect" unused objects?

Cloud Computing is similar but applied to virtual resources like servers, databases, storage, …

THE CLOUD COMPUTING DIFFERENCE

❶ IT Assets Become Programmable Resources

❷ **No need to know how to program resources**

- Rely on a higher level of managed services, such as Auto-scaling, Load balancers, …
- Deliver new solutions faster.
- Designed for scalability and high availability.

THE CLOUD COMPUTING DIFFERENCE

❶ IT Assets Become Programmable Resources

❷ No need to know how to program resources

❸ **Global, Available, and Unlimited Capacity**

- Whether you need to serve 1 user or 1 billion users.
- Whether you need to optimize network speed for US, Europe, Asia, etc.
- Move machines and data around the globe programmatically.
- Business Continuity.
- Disaster recovery.

AWS Global Infrastructure (May 2016)

THE CLOUD COMPUTING DIFFERENCE

❶ IT Assets Become Programmable Resources

❷ No need to know how to program resources

❸ Global, Available, and Unlimited Capacity

❹ **Security is Built-in**

- Native AWS security and encryption features can help achieve higher levels of data protection and compliance.
- Security policies built-into programmable resources.
- Continuous monitoring of configuration changes to your IT resources.
- Auditing is no longer periodic or manual, it becomes part of your continuous delivery pipeline.

AWS ARCHITECTURE DIAGRAMS

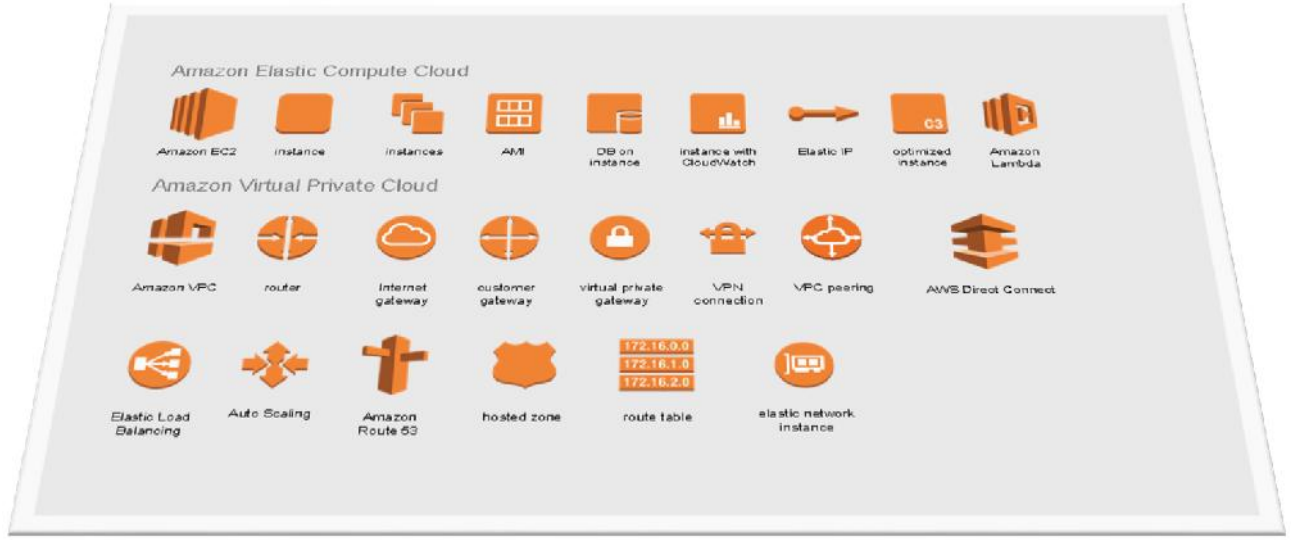
AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

- Compute & Networking
- Storage & Content Delivery
- Database
- Enterprise Applications
- Administration & Security
- Deployment & Management
- Application Services
- Analytics
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups

AWS ARCHITECTURE DIAGRAMS

AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

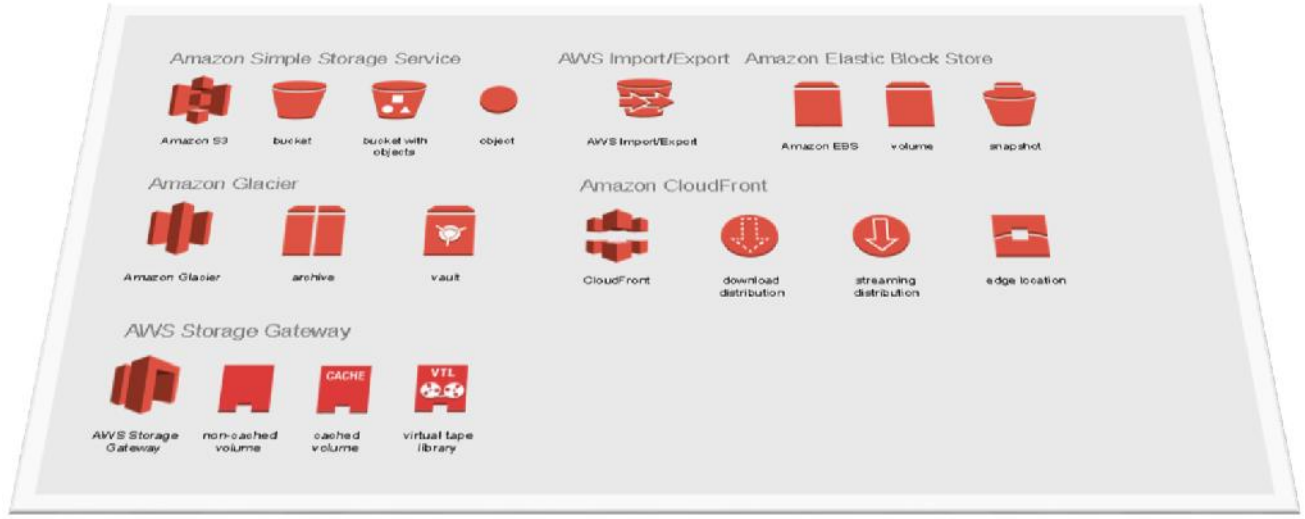
- **Compute & Networking**
- Storage & Content Delivery
- Database
- Enterprise Applications
- Administration & Security
- Deployment & Management
- Application Services
- Analytics
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups



AWS ARCHITECTURE DIAGRAMS

AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

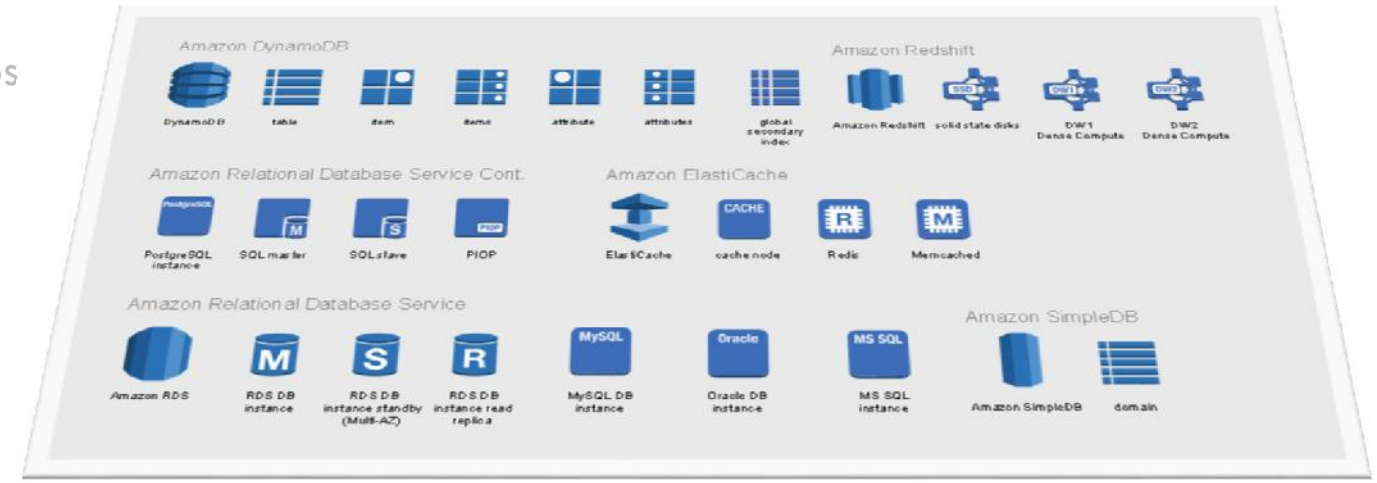
- Compute & Networking
- **Storage & Content Delivery**
- Database
- Enterprise Applications
- Administration & Security
- Deployment & Management
- Application Services
- Analytics
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups



AWS ARCHITECTURE DIAGRAMS

AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

- Compute & Networking
- Storage & Content Delivery
- **Database**
- Enterprise Applications
- Administration & Security
- Deployment & Management
- Application Services
- Analytics
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups



AWS ARCHITECTURE DIAGRAMS

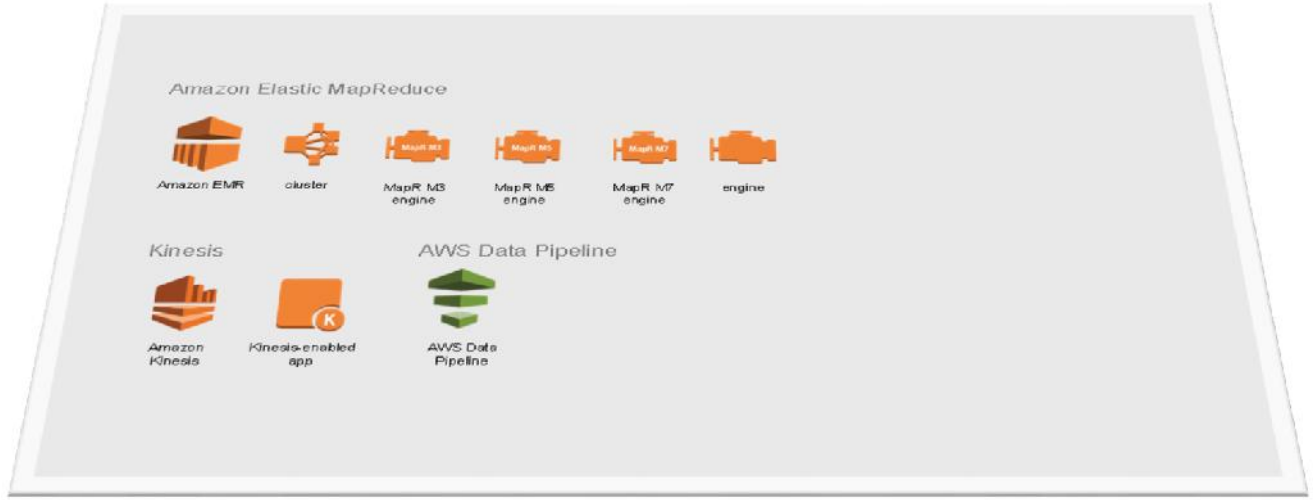
AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

- Compute & Networking
- Storage & Content Delivery
- Database
- Enterprise Applications
- Administration & Security
- Deployment & Management
- **Application Services**
- Analytics
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups

AWS ARCHITECTURE DIAGRAMS

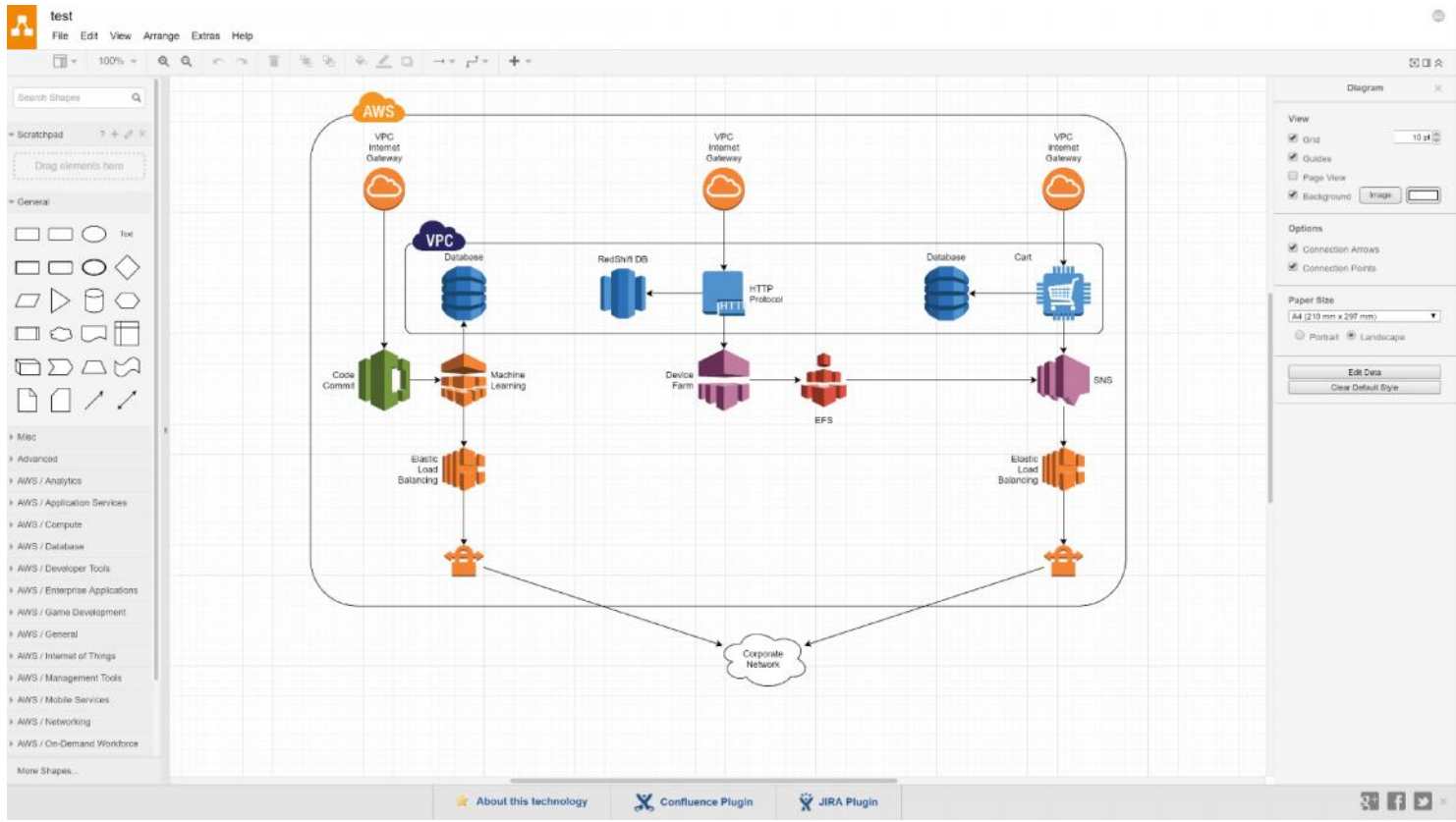
AWS architecture diagrams are a great way to communicate about your design, deployment and topology. In the following slides, the official collection of AWS Simple Icons v2.4 is used. These include:

- Compute & Networking
- Storage & Content Delivery
- Database
- Enterprise Applications
- Administration & Security
- Deployment & Management
- Application Services
- **Analytics**
- Mobile Services
- Non-Service Specific
- On-Demand Workforce
- SDKs
- Groups



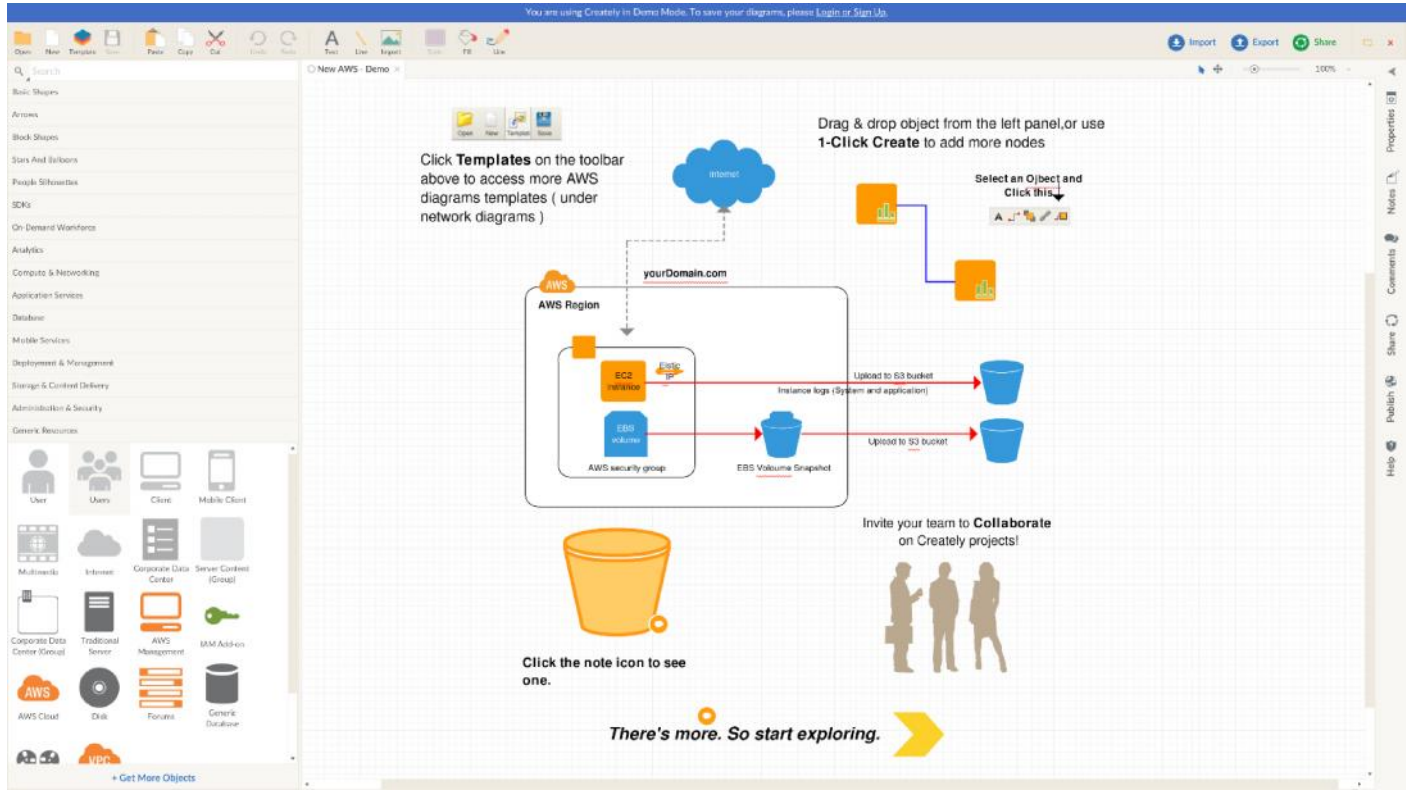
AWS ARCHITECTURE DIAGRAMS
TOOLS

draw.io



AWS ARCHITECTURE DIAGRAMS
TOOLS

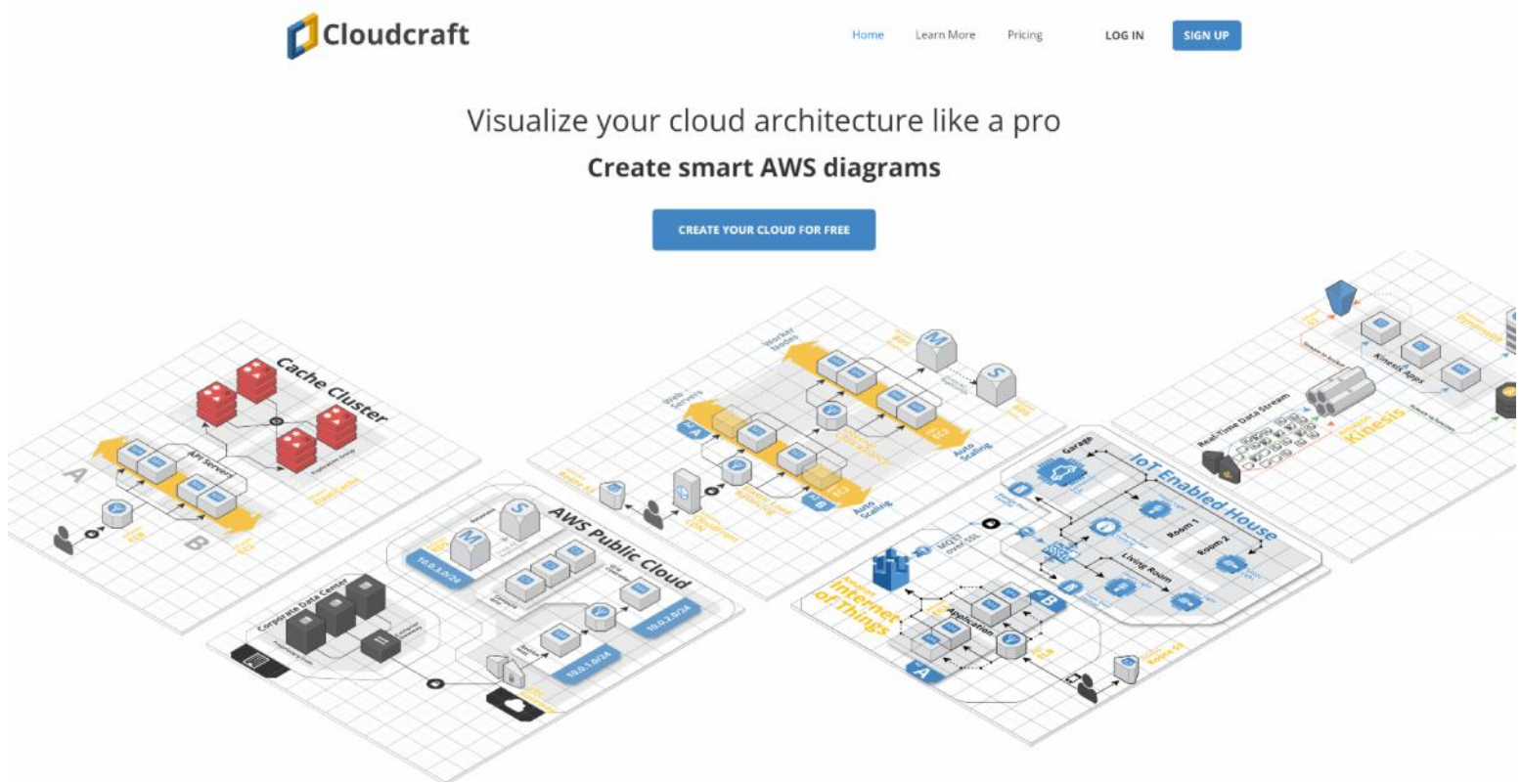
creately.com



AWS ARCHITECTURE DIAGRAMS
TOOLS

Cloudcraft

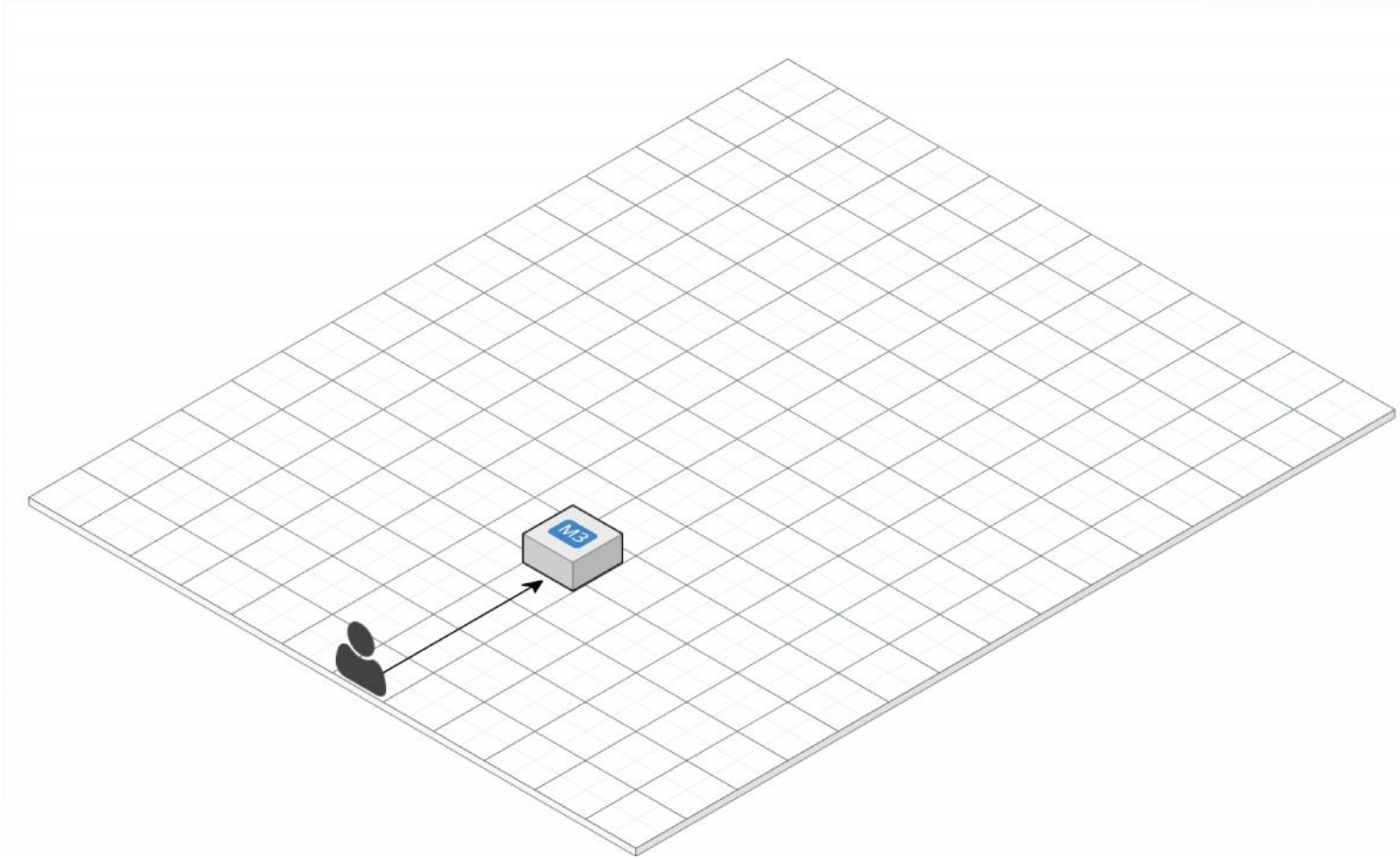
cloudcraft.co



DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Vertically

- 1 single EC2 instance
- Type: M3
- Size: Medium

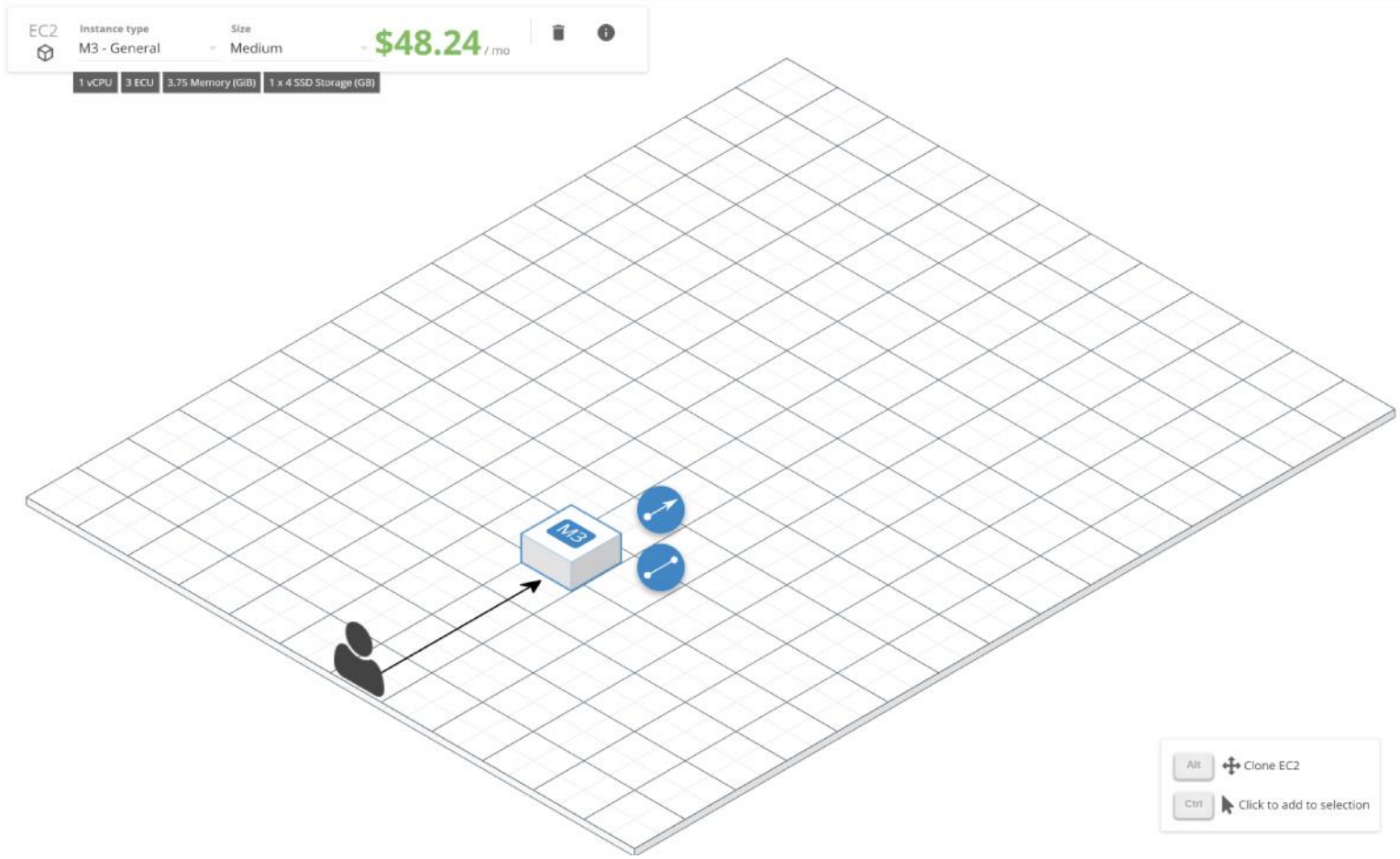


DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Vertically

- 1 single EC2 instance
- Type: M3
- Size: Medium
- COST: $48.24/month

Cheap but not large enough.

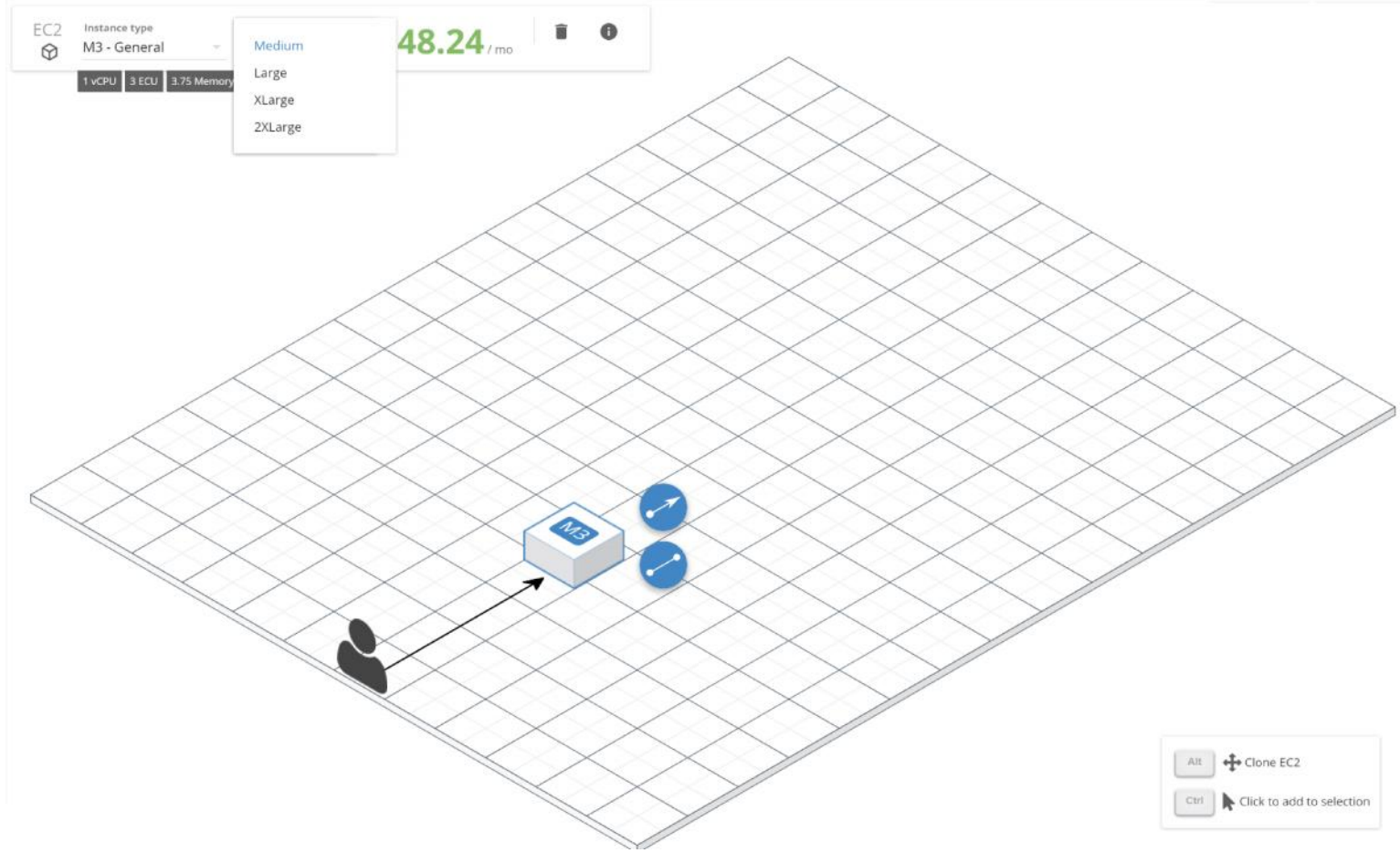


DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Vertically

- 1 single EC2 instance
- Type: M3
- Size: Medium

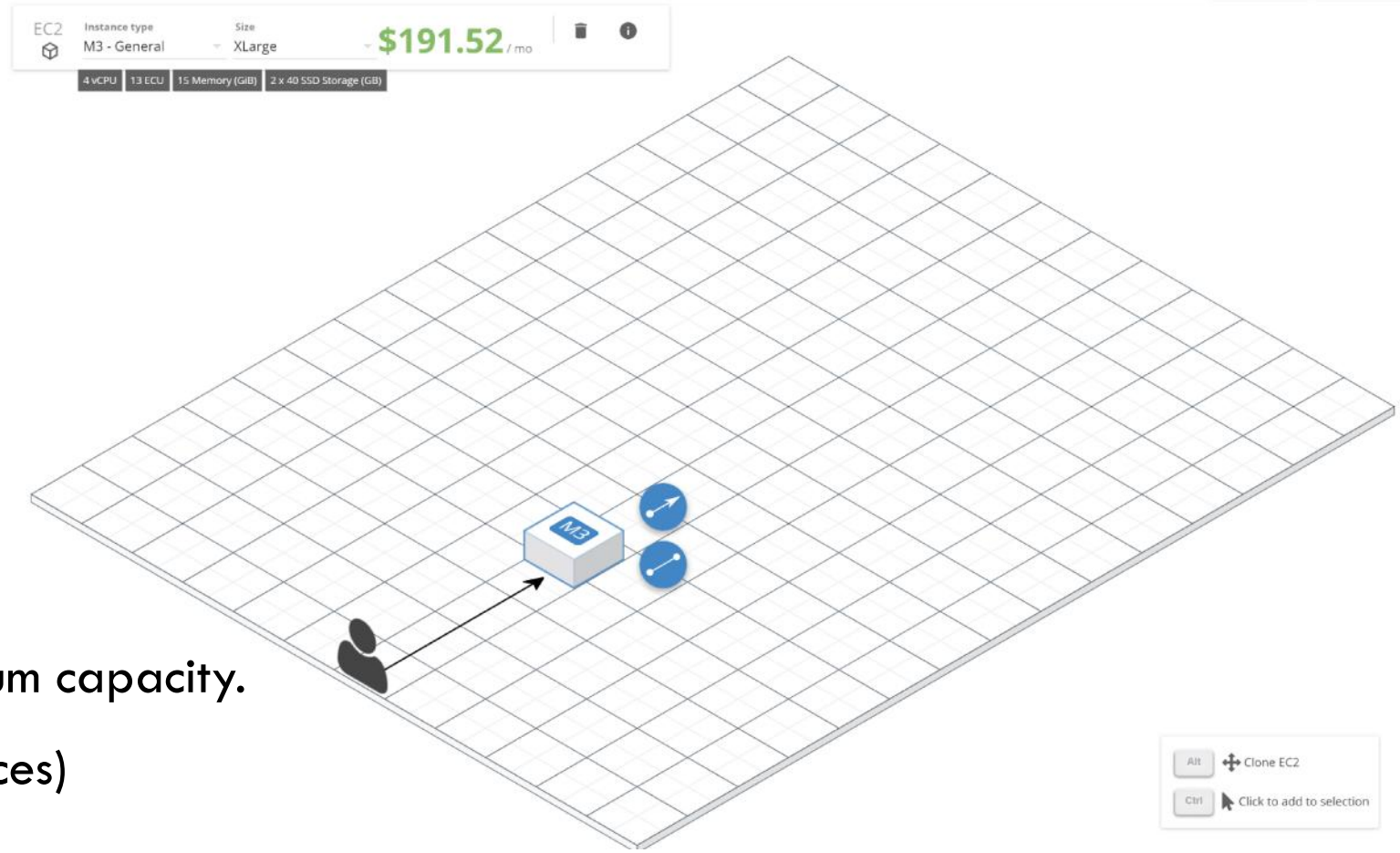
Selecting higher specification.



DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Vertically

- 1 single EC2 instance
- Type: M3
- Size: Xlarge
- COST: $191.52/month



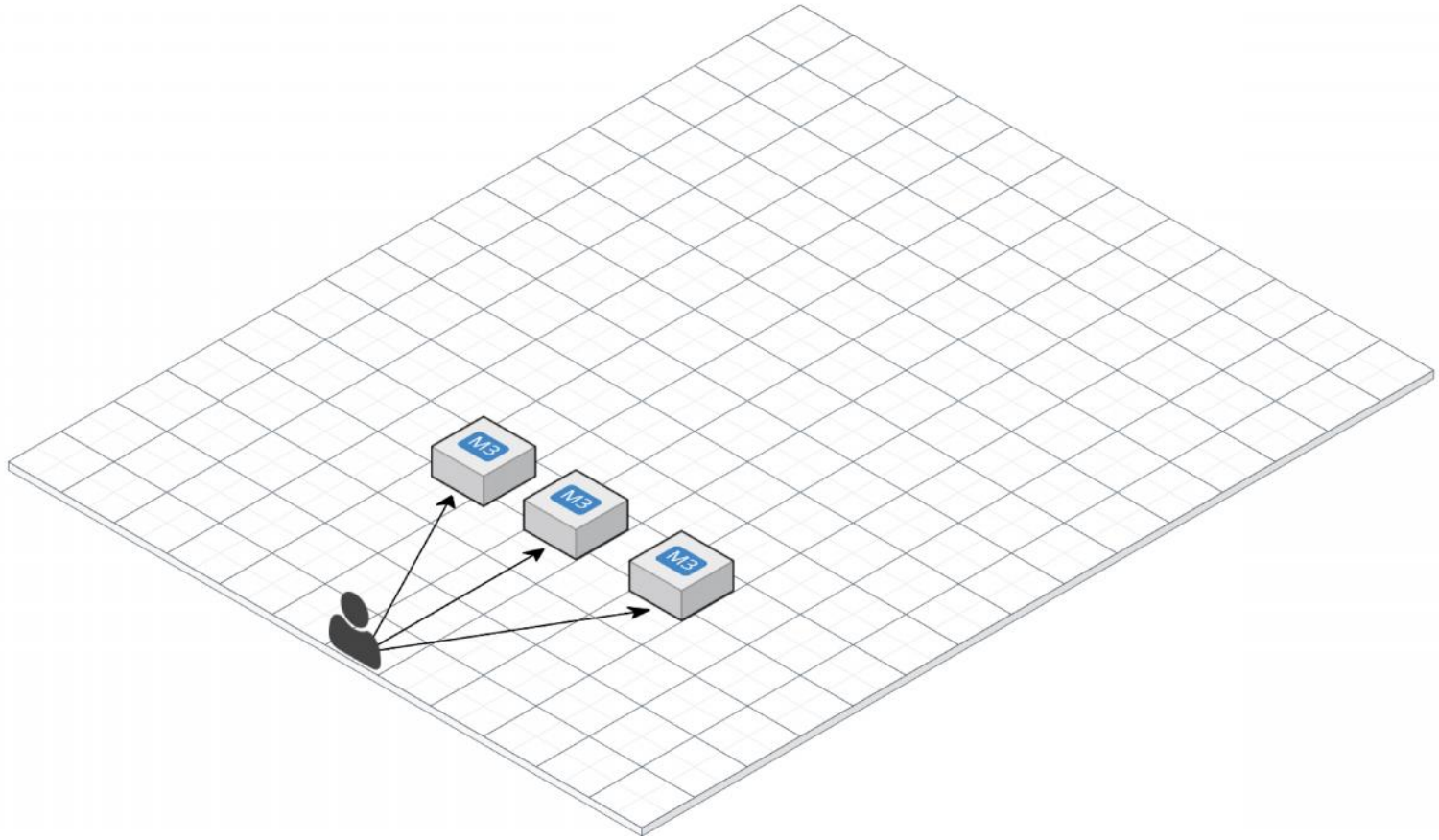
Still cheap but there is a maximum capacity.

(2Xlarge for M3-General instances)

DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Horizontally

- Multiple EC2 instances
- Type: M3
- Size: Medium



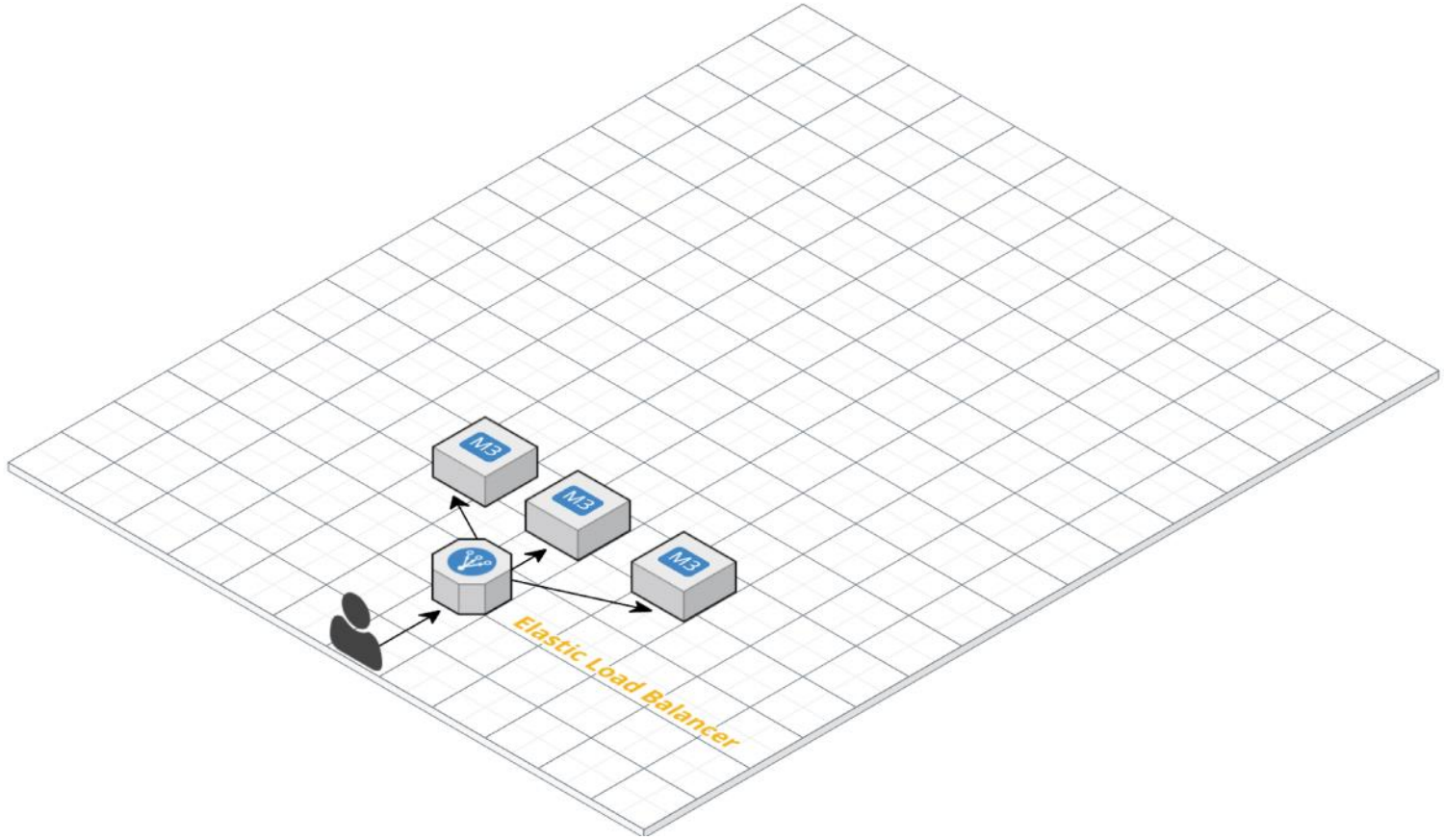
DESIGN PRINCIPLES FOR AWS
SCALABILITY

Scaling Horizontally

- Multiple EC2 instances
- Type: M3
- Size: Medium

+ Elastic Load Balancer

Unlimited capacity,
simply add more instances.



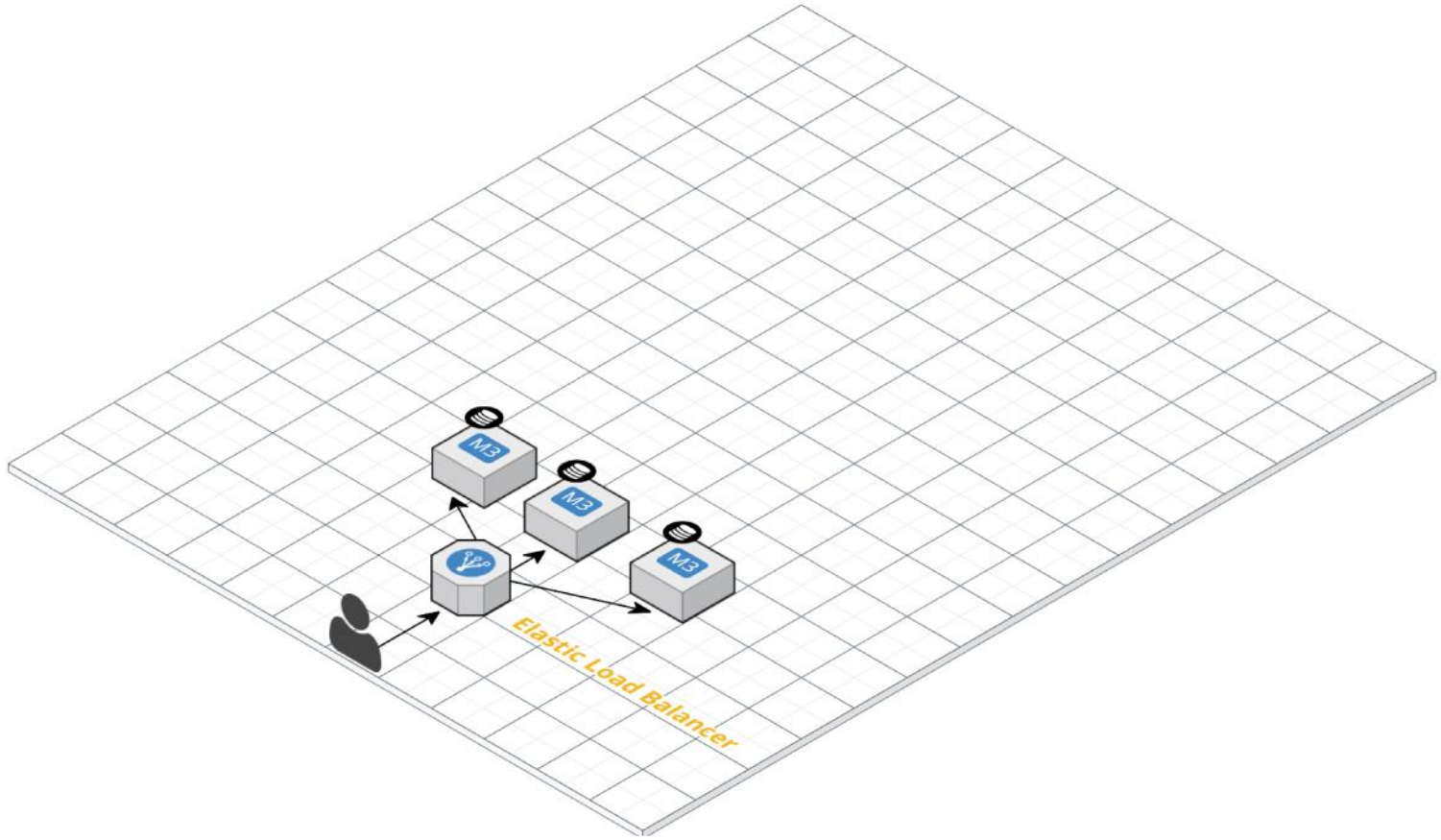
DESIGN PRINCIPLES FOR AWS **SCALABILITY**

Scaling Horizontally

- Multiple EC2 instances
- Type: M3
- Size: Medium

+ Elastic Load Balancer

What about Databases?



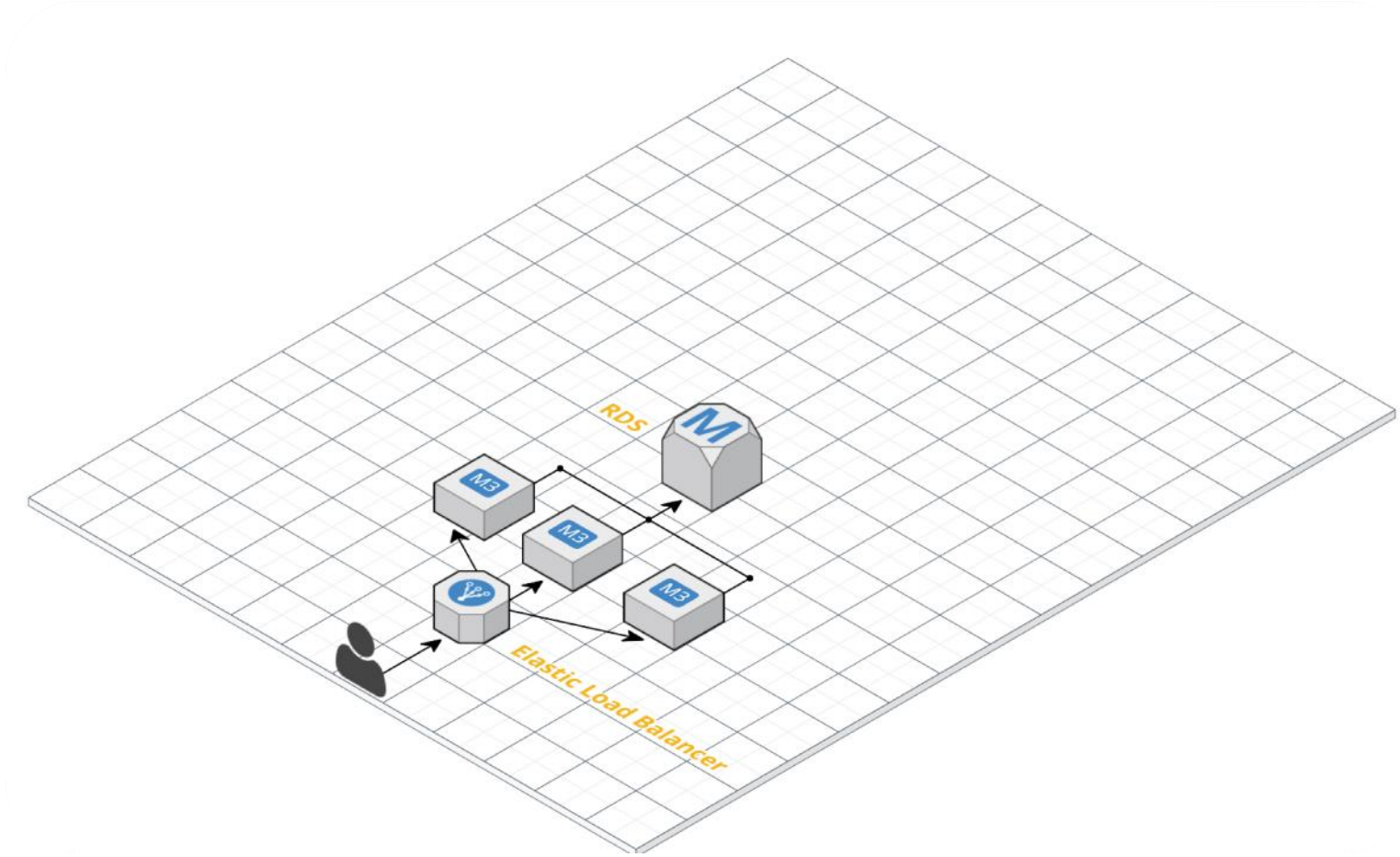
DESIGN PRINCIPLES FOR AWS **SCALABILITY**

Scaling Horizontally

- Multiple EC2 instances
- Type: M3
- Size: Medium

+ Elastic Load Balancer

+ Relational Database Service (RDS)

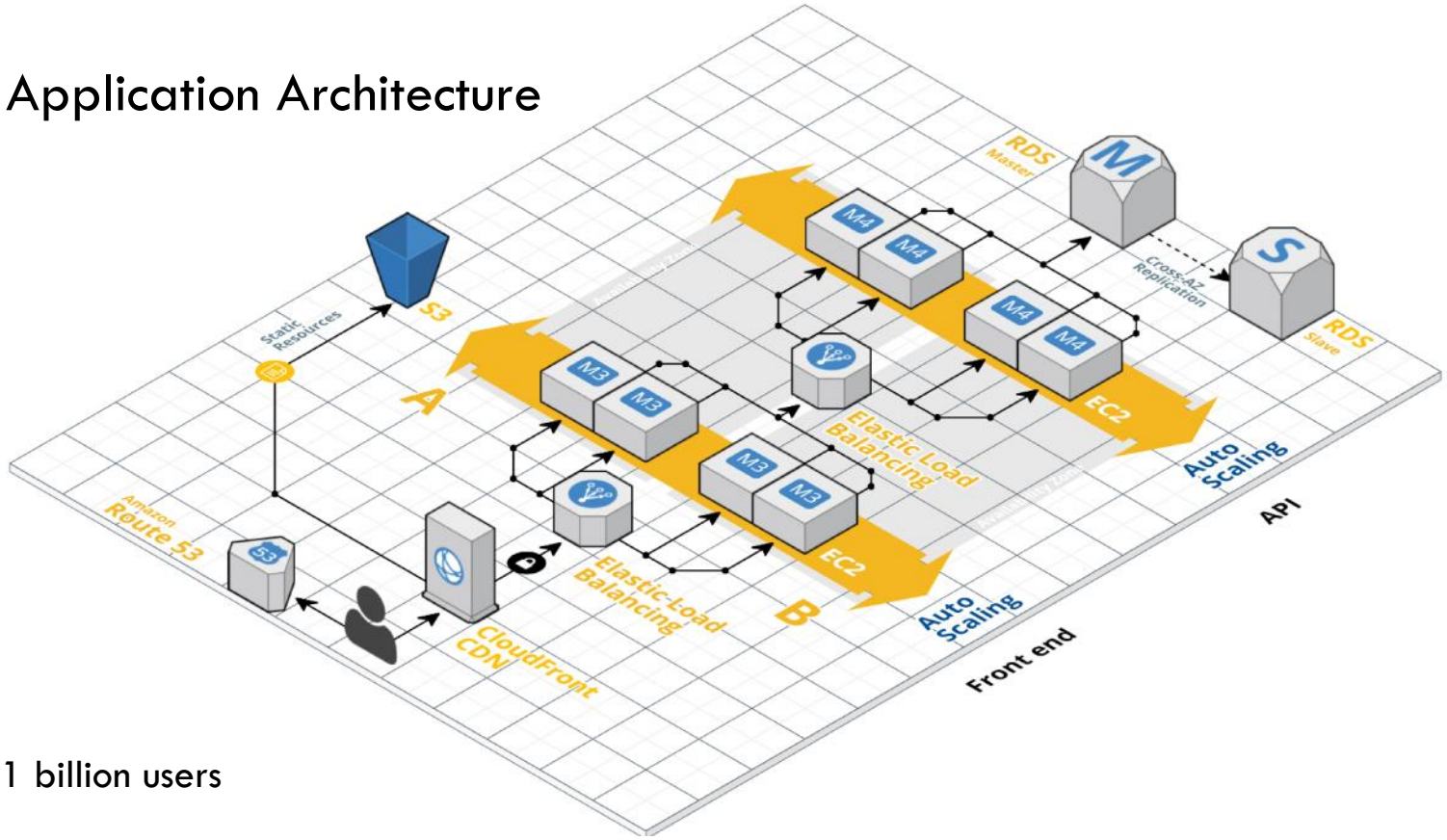


DESIGN PRINCIPLES FOR AWS **SCALABILITY**

A full example of Scalable Web Application Architecture

- Total minimum cost : $980.64/month

That's a good price for
a fully scalable deployment
that can serve 1 user
or 1 billion users*



* cost would sky rocket when scaled to serve 1 billion users

1 DESIGN PRINCIPLES FOR AWS
SCALABILITY AT ALL LEVELS

Scalable CDN
Content Delivery Network using AWS CloudFront.

Scalable DNS
Domain Name System web service using AWS Route 53.

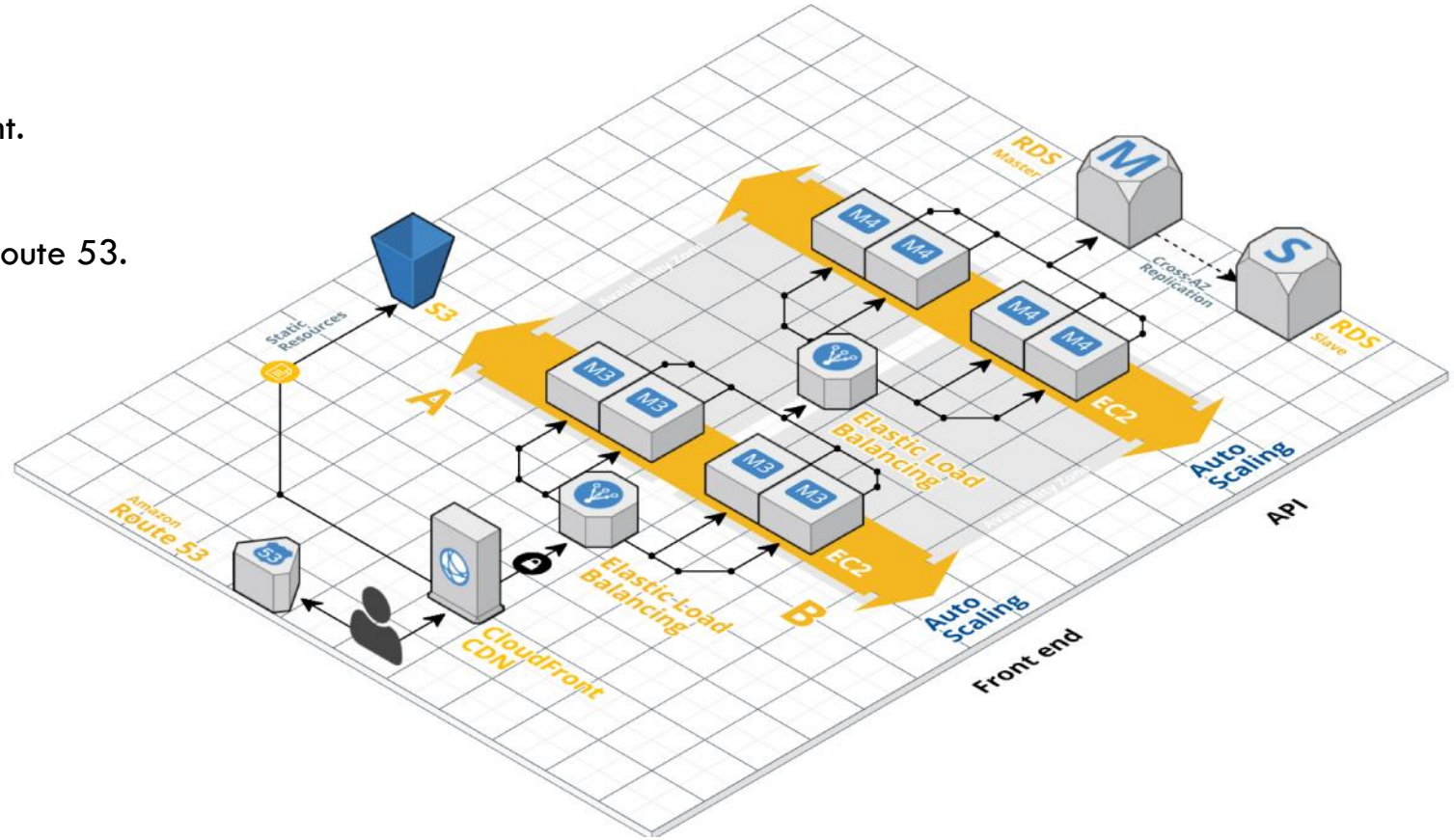
Scalable Load Balancer using ELB

Scalable Front end component
Elastic Compute Cloud (EC2) with Auto Scaling

Scalable API component
Elastic Compute Cloud (EC2) with Auto Scaling

Scalable DB
RDS with multi-AZ Replication

Scalable storage for static files
Simple Storage Service (S3)



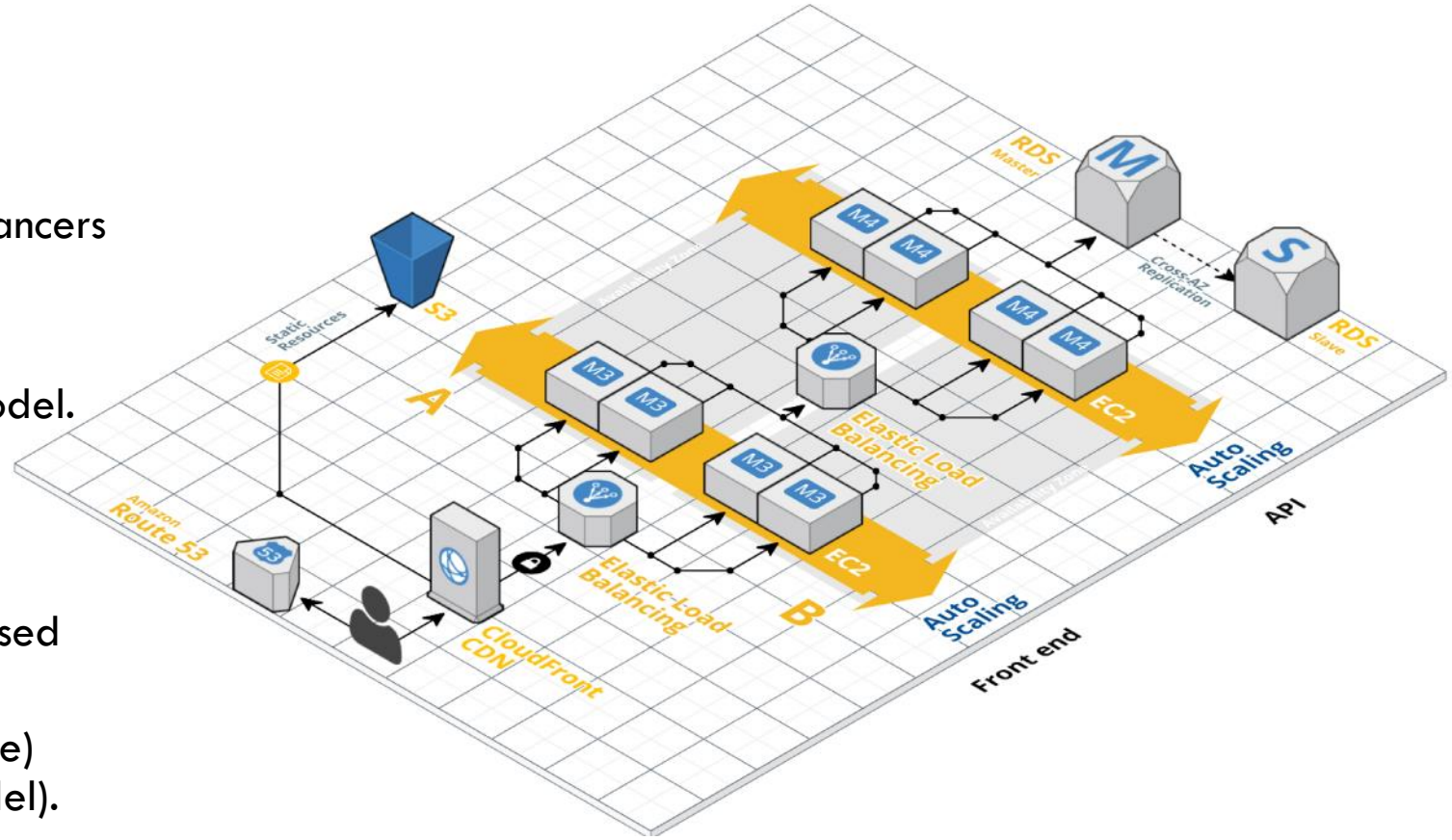
DESIGN PRINCIPLES FOR AWS
2 LOOSE COUPLING

Statefull components as services

- Example: Front end.
- Service Discovery via Elastic Load Balancers as stable endpoints.
- Use of Sticky Sessions.
- Load balancers implement the Push model.

Stateless components as services

- Example: Backend API.
- Load balancers (Push Model) can be used but not best
- Ideally use SQS (Simple Queue Service) for Asynchronous Integration (Pull Model).



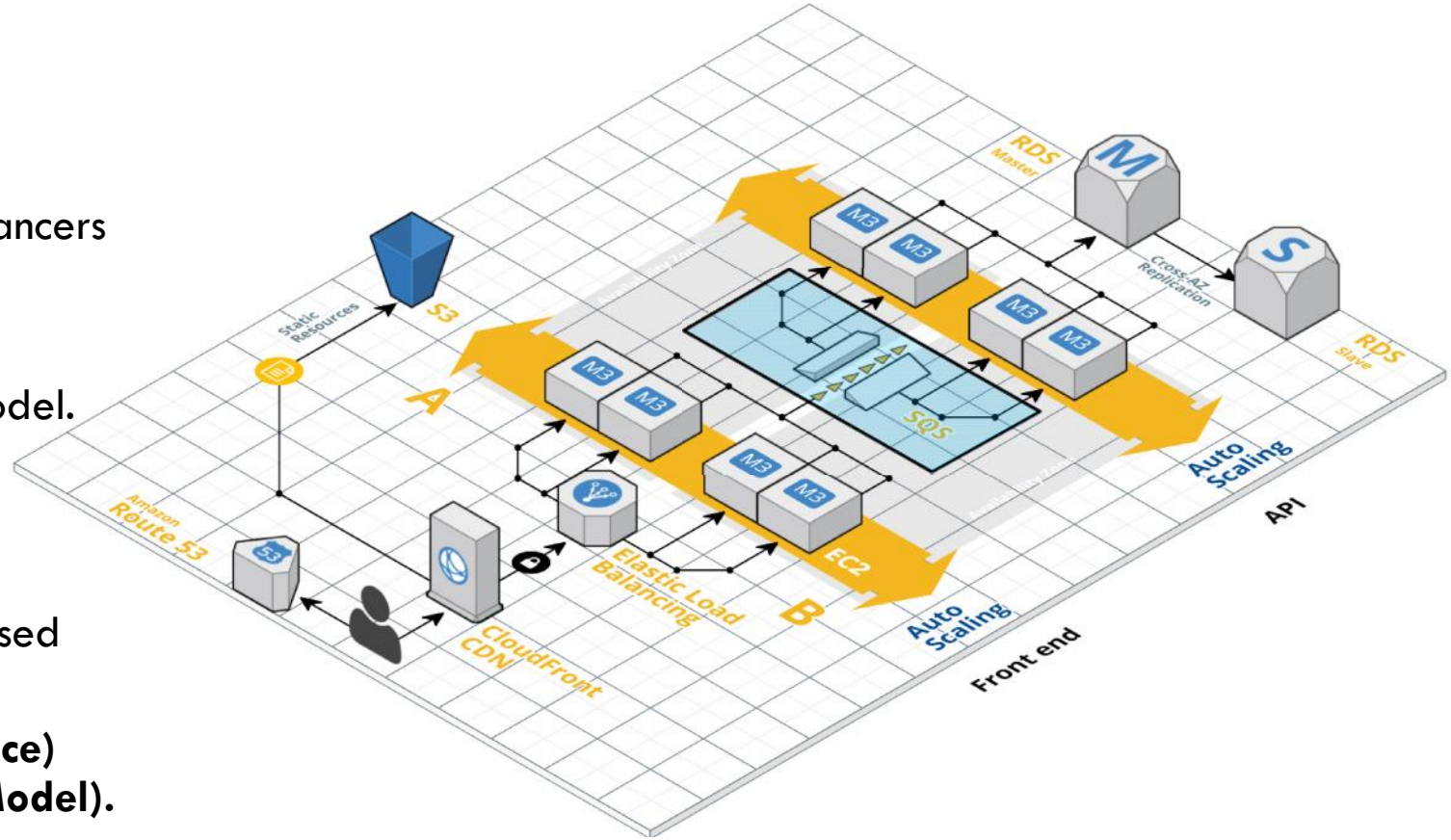
2 DESIGN PRINCIPLES FOR AWS LOOSE COUPLING

Statefull components as services

- Example: Front end.
- Service Discovery via Elastic Load Balancers as stable endpoints.
- Use of Sticky Sessions.
- Load balancers implement the Push model.

Stateless components as services

- Example: Backend API.
- Load balancers (Push Model) can be used but not best
- **Ideally use SQS (Simple Queue Service) for Asynchronous Integration (Pull Model).**



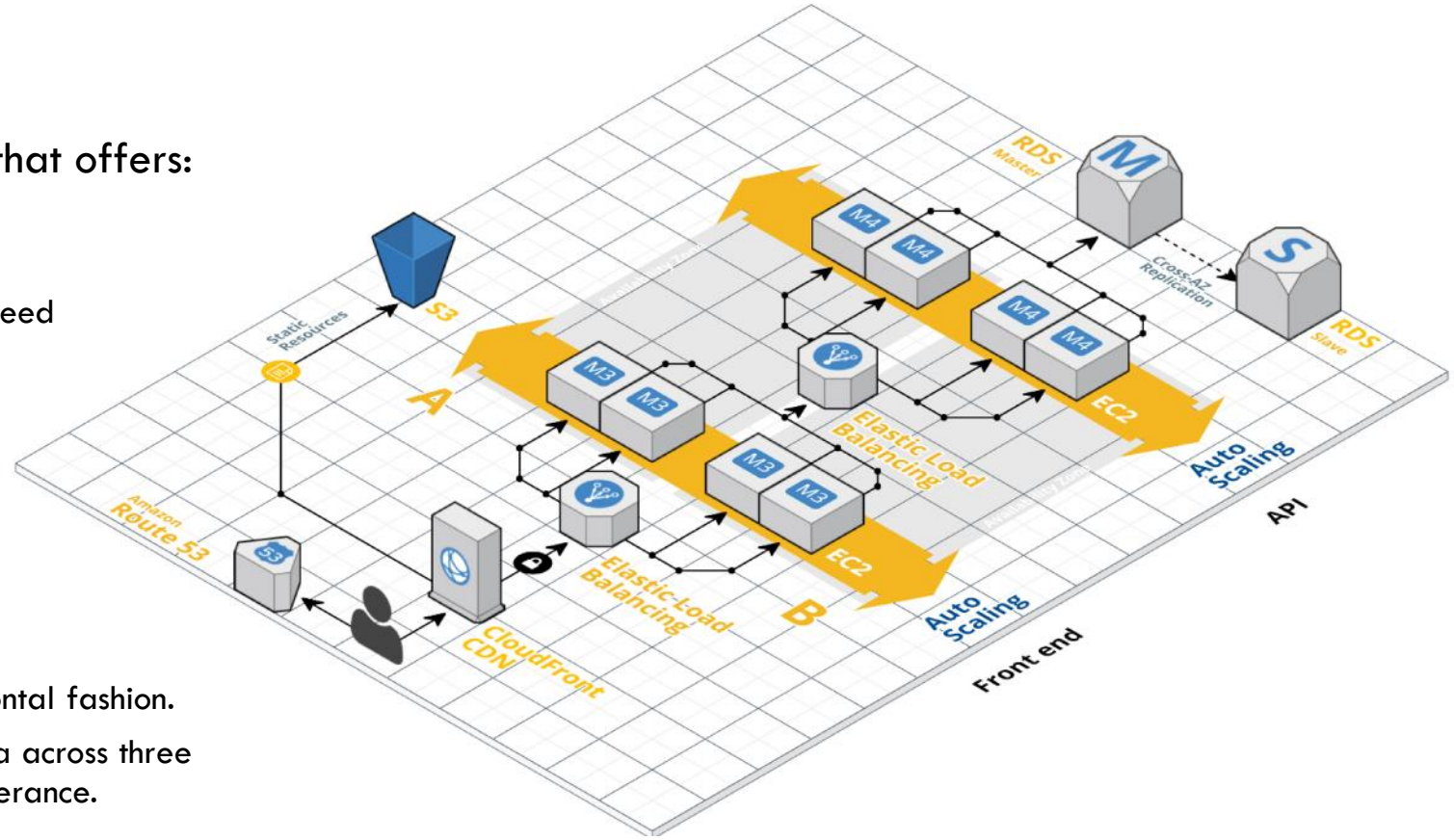
3 DESIGN PRINCIPLES FOR AWS
CHOSE THE RIGHT DATABASE(S)

Relational Databse

- RDS is a MySQL compatible databse that offers:
  - Scalability (both vertically and horizontally).
  - High Availability (RDS Multi-AZ deployment). Automatic failover to the standby without the need for manual administrative intervention.

NoSQL Databases

- DynamoDB is a NoSQL database.
  - Suitable if your application primarily indexes and queries data with no need for joins or complex transactions.
  - Scale both the reads and the writes in a horizontal fashion.
  - High Availability: synchronously replicates data across three facilities in an AWS region to provide fault tolerance.

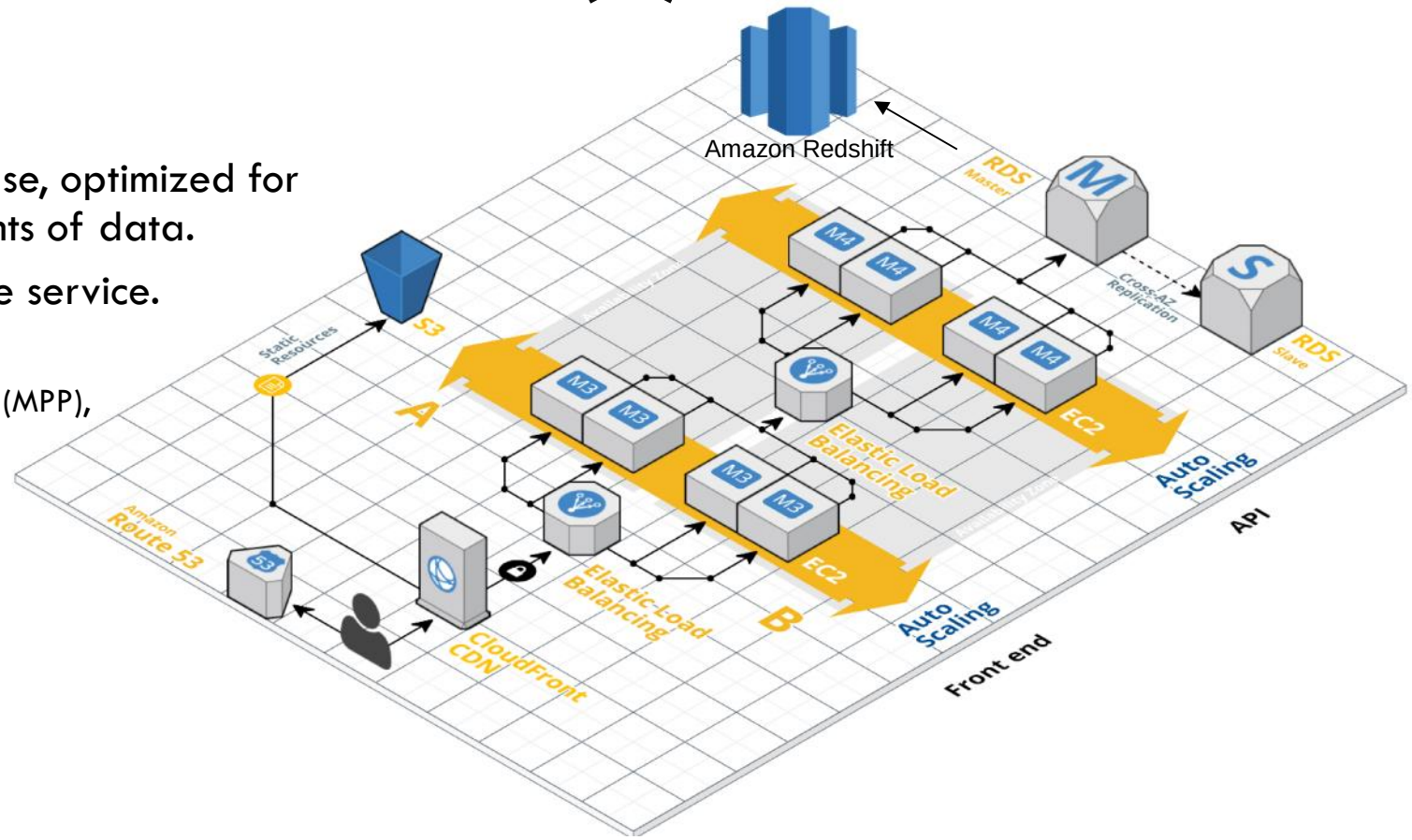


3 DESIGN PRINCIPLES FOR AWS
CHOSE THE RIGHT DATABASE(S)

Data Warehouse

- Specialized type of relational database, optimized for analysis and reporting of large amounts of data.
- Redshift is a managed data warehouse service.
  - SQL-based.
  - Scalability using massively parallel processing (MPP), columnar data storage, and targeted data compression encoding schemes.
  - High Availability with multi-node clusters in which data written to a node is automatically replicated to other nodes within the cluster.

    + backed up to Amazon S3.

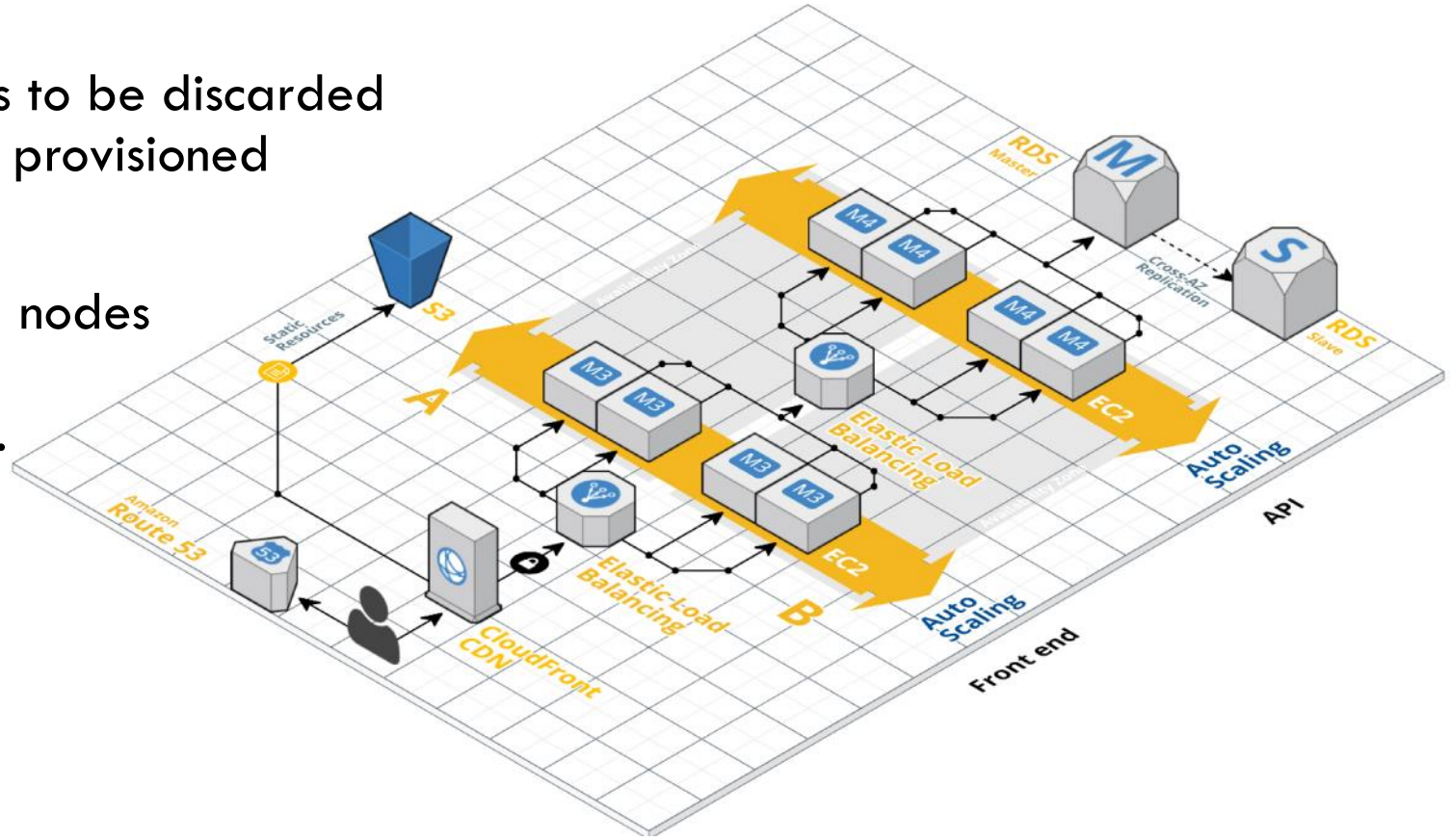


4 DESIGN PRINCIPLES FOR AWS
DISPOSABLE RESOURCES INSTEAD OF FIXED SERVERS

Auto Scaling allows EC2 instances to be discarded when not used and new instances provisioned in seconds to meet the demand.

RDS database scales into several nodes and any node can be discarded without impact on DB availability.

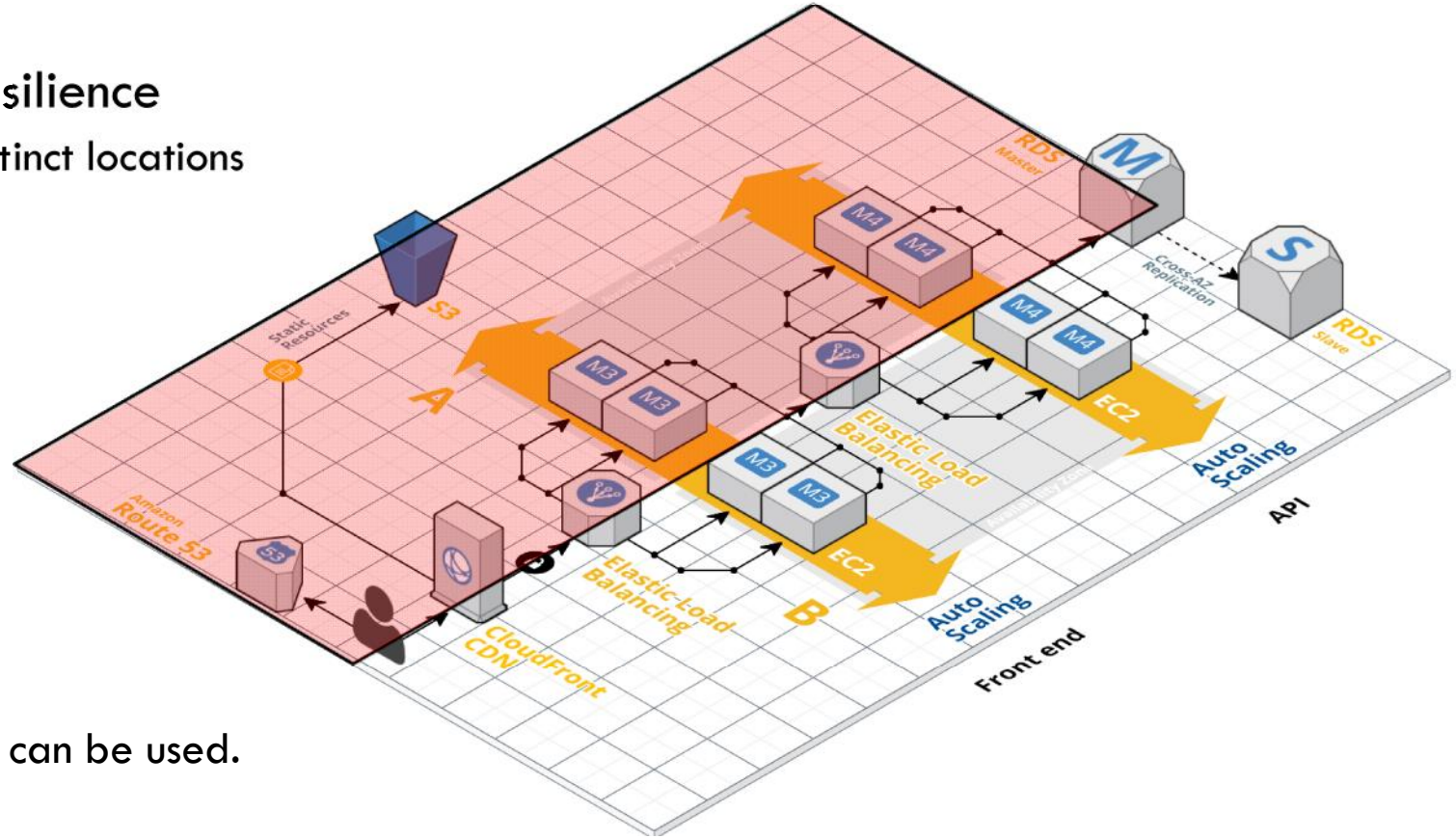
Everything else is a service.



5 DESIGN PRINCIPLES FOR AWS
REMOVING SINGLE POINTS OF FAILURE

Automated Multi-Data Center Resilience

- Each AWS region contains multiple distinct locations called Availability Zones (AZ).
- Each AZ is engineered to be isolated from failures in other AZ.
- If AZ "A" fails, failover is automated and all requests are routed to the working AZ "B".

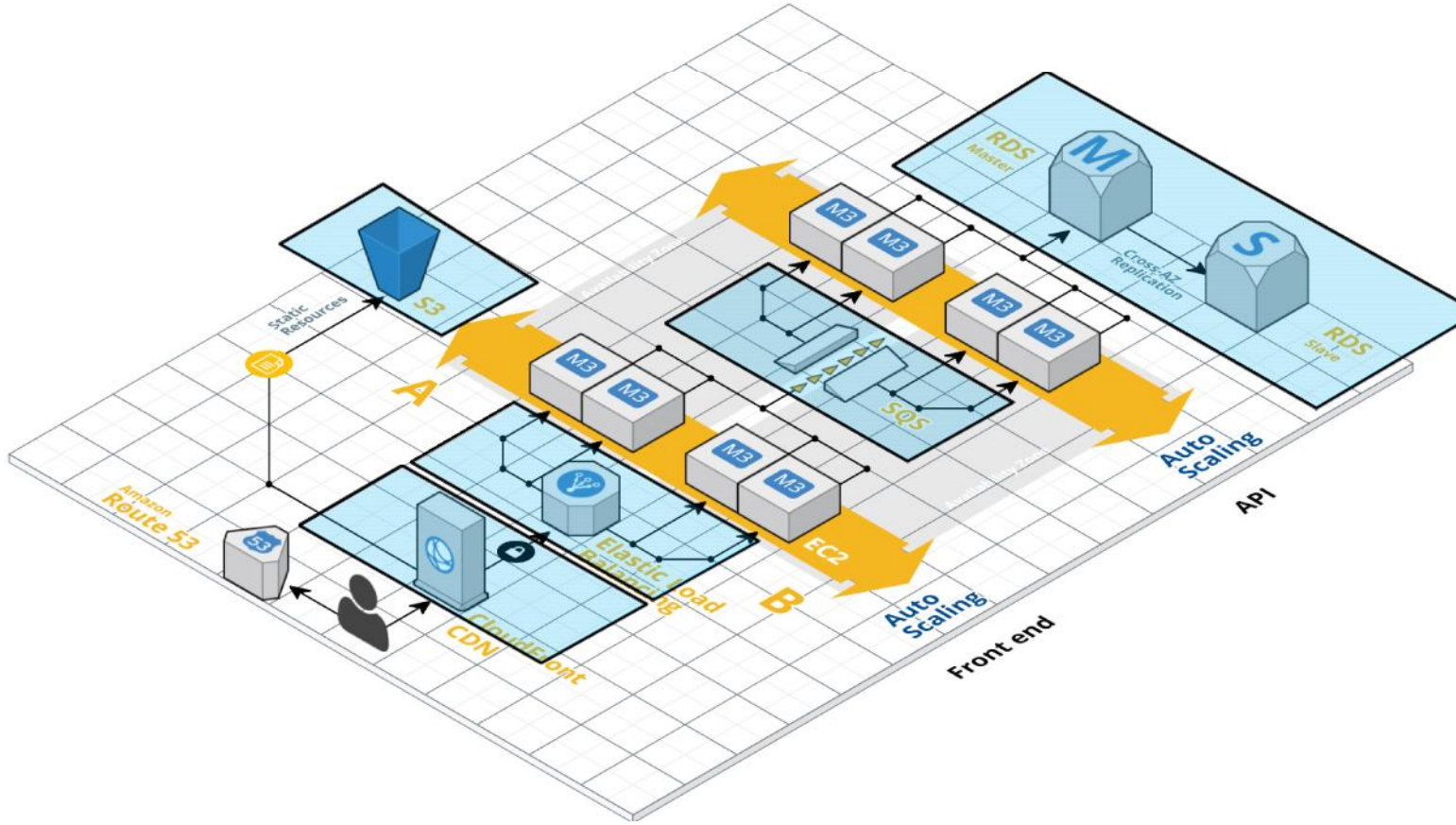


Note: Multiple AZs and multiple regions can be used.

6 DESIGN PRINCIPLES FOR AWS
SERVICES, NOT SERVERS (AS MUCH AS POSSIBLE)

Use Managed Services

- Simple Queue Service (SQS).
- CloudFront for content delivery.
- Elastic Load Balancers.
- RDS.
- DynamoDB.
- CloudSearch.
- Simple Email Service (SES).
- S3.
- …

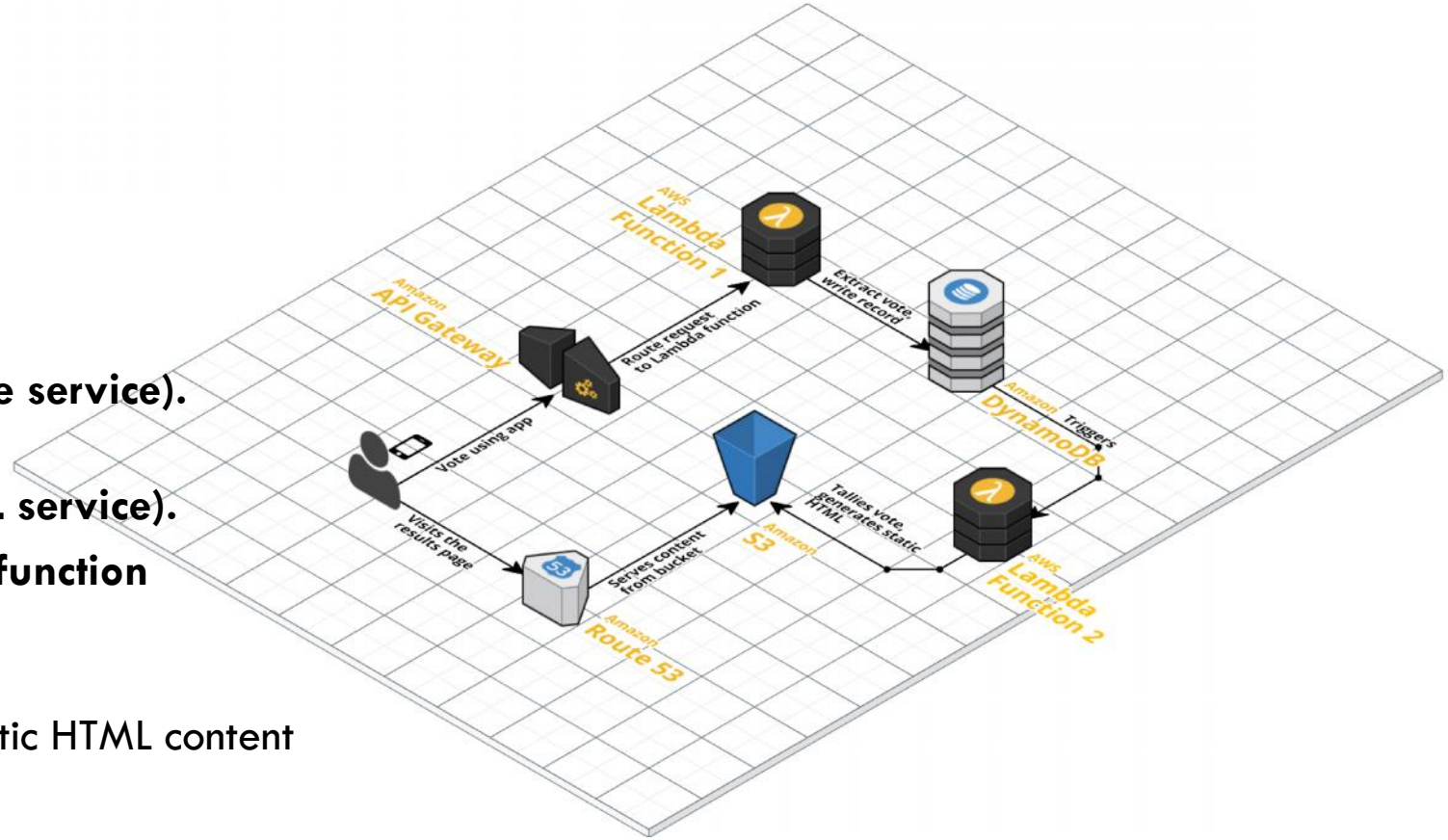


6 DESIGN PRINCIPLES FOR AWS SERVICES, NOT SERVERS (100%)

Serverless Architectures

Example: Voting mobile app.

- Voting app on mobile uses **Amazon API Gateway** to vote.
- Vote request is routed to an **AWS Lambda function (compute service).**
- **AWS Lambda function** extracts info and record it into **DynamoDB (NoSQL service).**
- **DynamoDB** triggers an**AWS Lambda function** to generate static HTML and store it into **S3 (file storage service).**
- **Route 53 (DNS service)** serves the static HTML content from **S3** to the user.



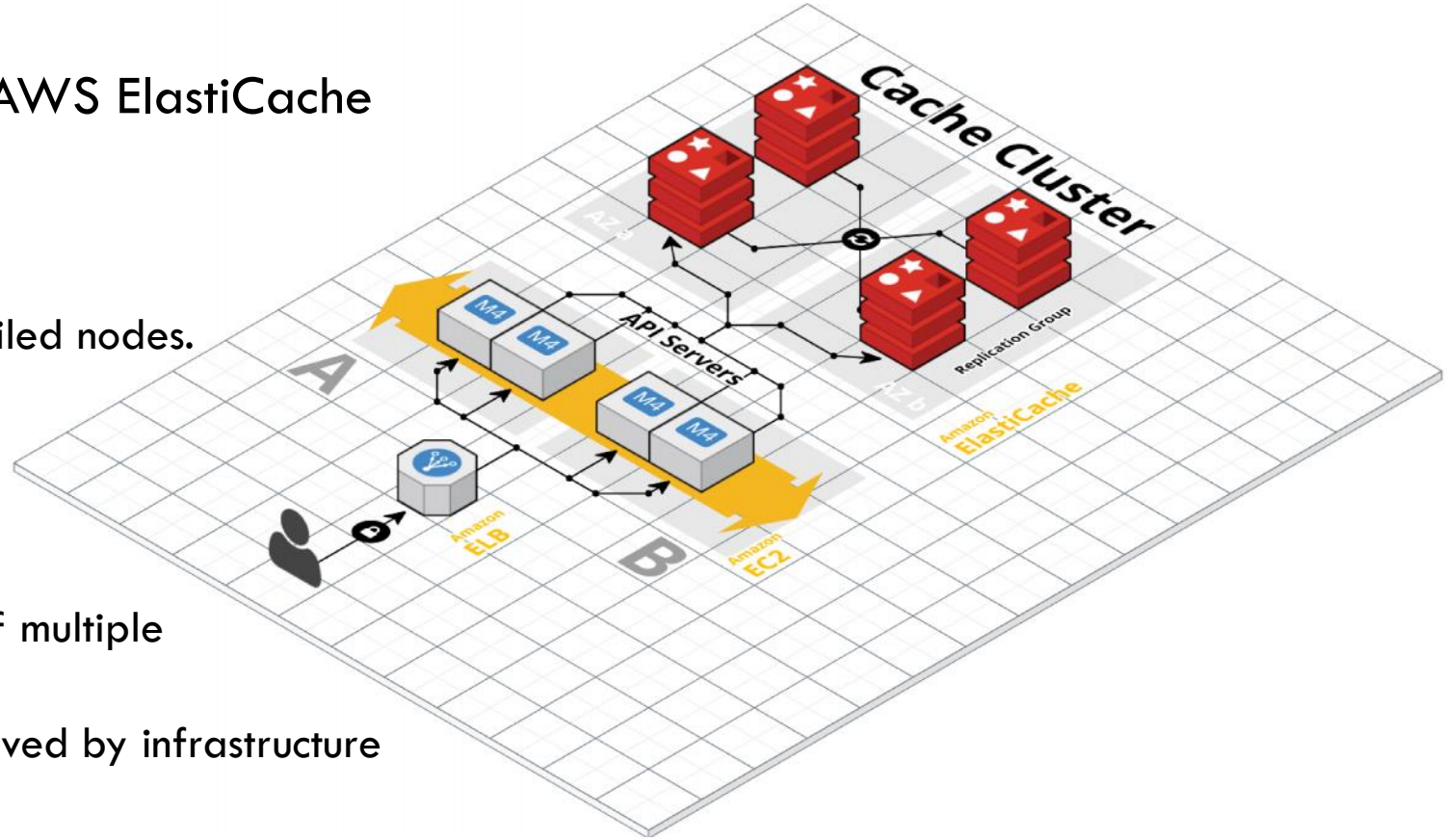
7 DESIGN PRINCIPLES FOR AWS CACHING

Application Data Caching using AWS ElastiCache

- In-memory caching engines:
  - Memcached (objects).
  - Redis (key-value store).
- Automatically detects and replaces failed nodes.
- Faster than disk.
- Scale vertically or horizontally.

Edge Caching using CloudFront

- Content Delivery Network consisting of multiple edge locations around the world.
- Edge caching allows content to be served by infrastructure that is closer to viewers.



DESIGN PRINCIPLES FOR AWS
8 OPTIMIZE FOR COST

Select the right type/size for your instances.

- General Purpose vs Compute Optimized vs Memory Optimized vs Storage Optimized.
- Burstable Performance Instances or fixed performance instances.
- Large vs Xlarge vs 2Xlarge vs 4Xlarge vs 10Xlarge.
- Many small instances vs fewer large instances.

Rely on Auto-Scaling to always fit the demand and pay for what you need.

Take Advantage of the Variety of Purchasing Options.

- Reserved Capacity.
- On-Demand vs Spot Instances (Bidding strategy).
- Mix On-Demand and Spot Instances.

9 DESIGN PRINCIPLES FOR AWS SECURITY

Use VPC (Virtual Private Cloud).

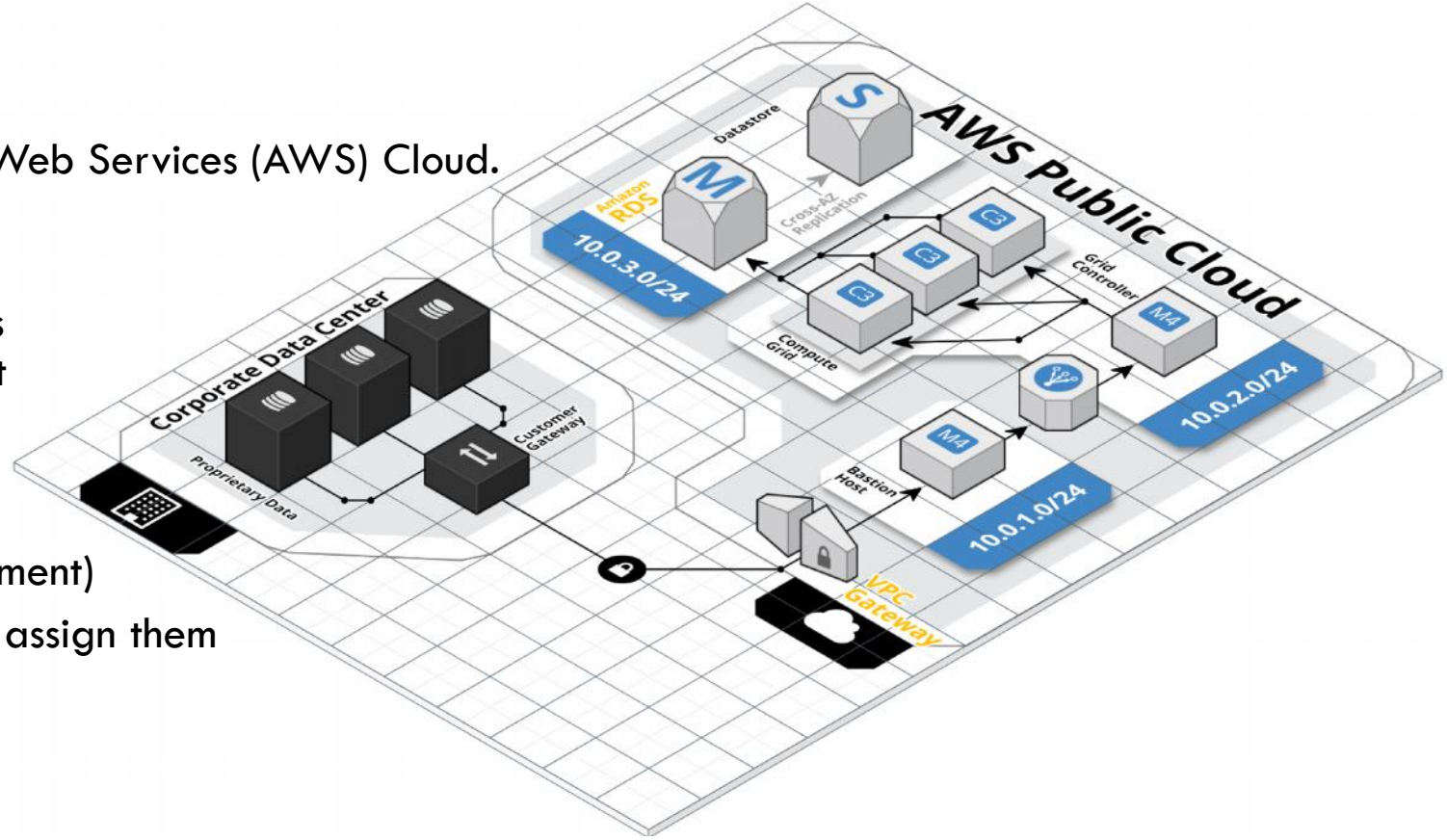
A logically isolated section of Amazon Web Services (AWS) Cloud.

Use a Bastion Host.

A special purpose server instance that is designed to be the primary access point and acts as a proxy to your other EC2 instances.

Use IAM (Identity and Access Management)

to define a granular set of policies and assign them to users, groups, and AWS resources.



9 DESIGN PRINCIPLES FOR AWS SECURITY

For Web Applications.

Use WAF (Web Application Firewall).

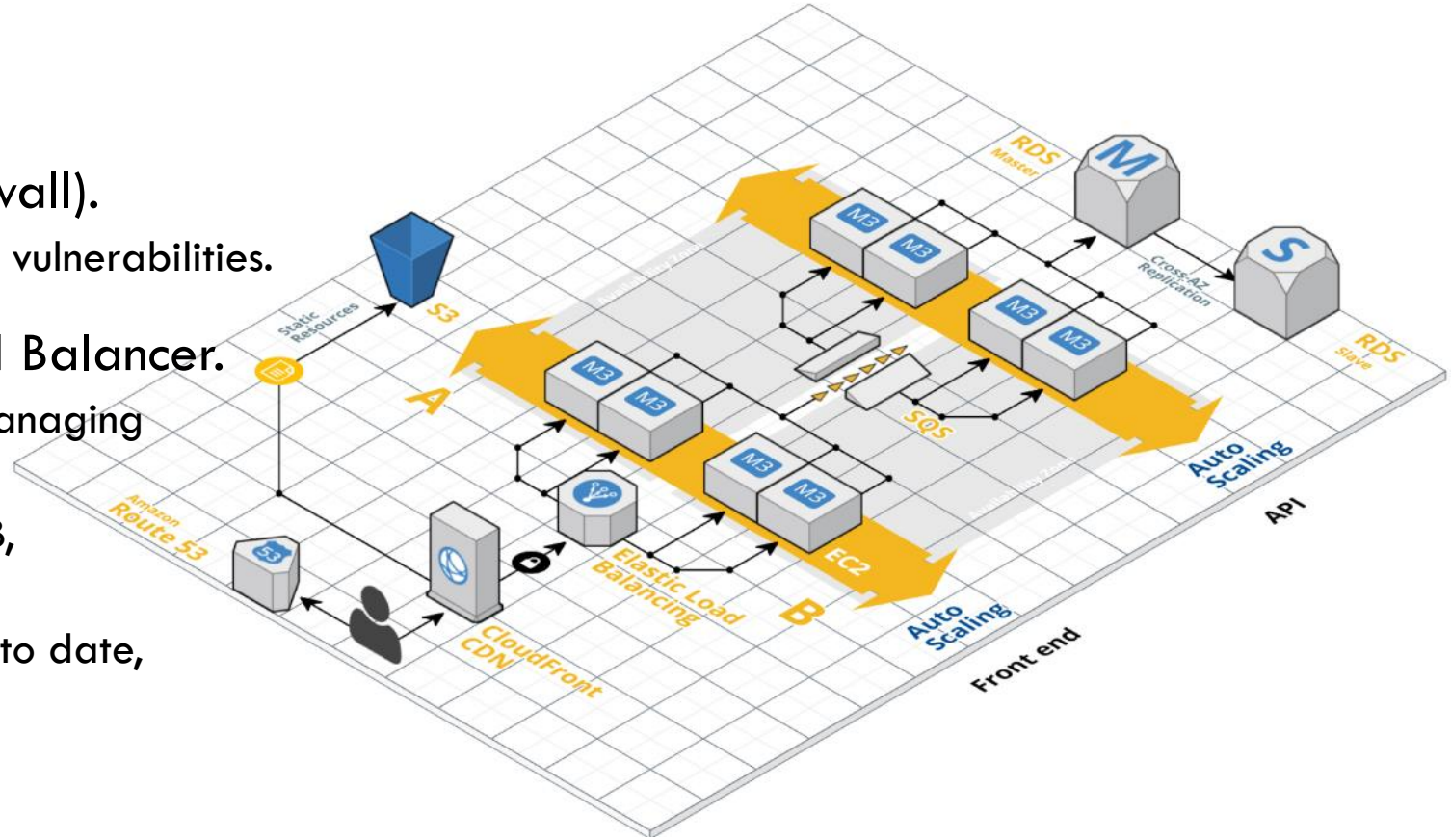
Protects against SQL injection and other vulnerabilities.

Setting up SSL on an Elastic Load Balancer.

Allows to offload your instances from managing SSL encryption/decryption.

Your SSL certificates are safe, within ELB, not within your instances.

Cypher suite configuration is always up to date, upgraded by Amazon when necessary (in case of new vulnerability).



SUMMARY OF DESIGN PRINCIPLES FOR AWS

❶ Scalability at all levels.

❷ Loose Coupling.

❸ Chose the right database(s).

❹ Disposable Resources Instead of Fixed Servers.

❺ Removing Single Points of Failure.

❻ Services, Not Servers (as much as possible).

❼ Caching.

❽ Optimize for Cost.

❾ Security.

LEARN MORE AND KEEP UP TO DATE

White Paper: Architecting for the Cloud - AWS Best Practices
https://aws.amazon.com/whitepapers/architecting-for-the-aws-cloud-best-practices/

Amazon regularly introduces new services, so keep up to date
https://aws.amazon.com/new/

AWS Free Tier (12 months free to get started)
https://aws.amazon.com/free/

Access these slides on

slideshare.net/guabtni

THANK YOU FOR YOUR ATTENTION

Dr Adnene Guabtni
Adnene.Guabtni@csiro.au