Database Architecture for SaaS

ARUL SHAJI | PRINCIPAL DATABASE CONSULTANT | ATLASSIAN

OUR USE OF POSTGRESQL

All of Atlassian’s relational database needs are served by PostgreSQL

Building a database platform that is flexible, scalable and secure, is key to a successful SaaS offering.

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Considerations

Data Isolation

Blast radius

Noisy neighbour

Scalable

Seperate or co-located

There are varying levels of data separation from physical to logical

Fundamentally about Multi tenancy

Directly translates into how we pack data belonging to multiple tenants.

There are contributing factors

Regulatory compliance, customer requirements/ demands can all play a part

Considerations

Data Isolation

Blast radius

Noisy neighbour

Scalable

How disruptive can a failure be

When something goes wrong, we want it to be as isolated as possible. ie., disruption is contained and impact is felt for the smallest number of customers

How quickly can we come back

This also has a bearing on how quickly we can recover from failures as well. RTO needs to be taken into account.

Considerations

Data Isolation

Blast radius

Noisy neighbour

Scalable

Consistent user experience

Ideally we want to give all users a consistent level of performance irrespective of the load on the system

A large tenant's activity should have no bearing on the experience of a smaller tenant

Maintaining an adequate level of service means that we should also be able to throttle users to stop them from hogging system resources

Considerations

Data Isolation

Blast radius

Noisy neighbour

Scalable

Should work the same for 10 or 10K tenants

Platform should be able to expand seamlessly as they grow

High level of automation

We cannot have a platform that has a high operational cost.

Minimal maintenance

Tooling that are specific to needs has to be built. Consider options to backup, patching, upgrade, etc.

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Single Server, Single tenant

Each tenant is placed in a completely isolated database server

Very expensive option

Could result in a lot of wasted resources and probably overkill

Multiple Databases, (or Schemas) Multiple tenants

There can be multiple database servers, each hosting multiple databases, or schemas with multiple tenants in each one of them.

When properly orchestrated, could be the ideal solution

Provides a good balance in having tenants of different profile to co-exist.

Single Database, Multiple tenants

Truly multi tenanted system. Underneath, these could be clustered and have multiple entry points.

Most requirements can still be achieved when designed and structured properly

Does place severe constraints on certain maintenance operations though

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Foundations

Cloud Native

Regions, AZs, Integrations and more

RDS PostgreSQL

Scalable, Low operational costs and flexible (really)

Apps and Tools

For monitoring, log analysis, debugging, etc.

Database per tenant

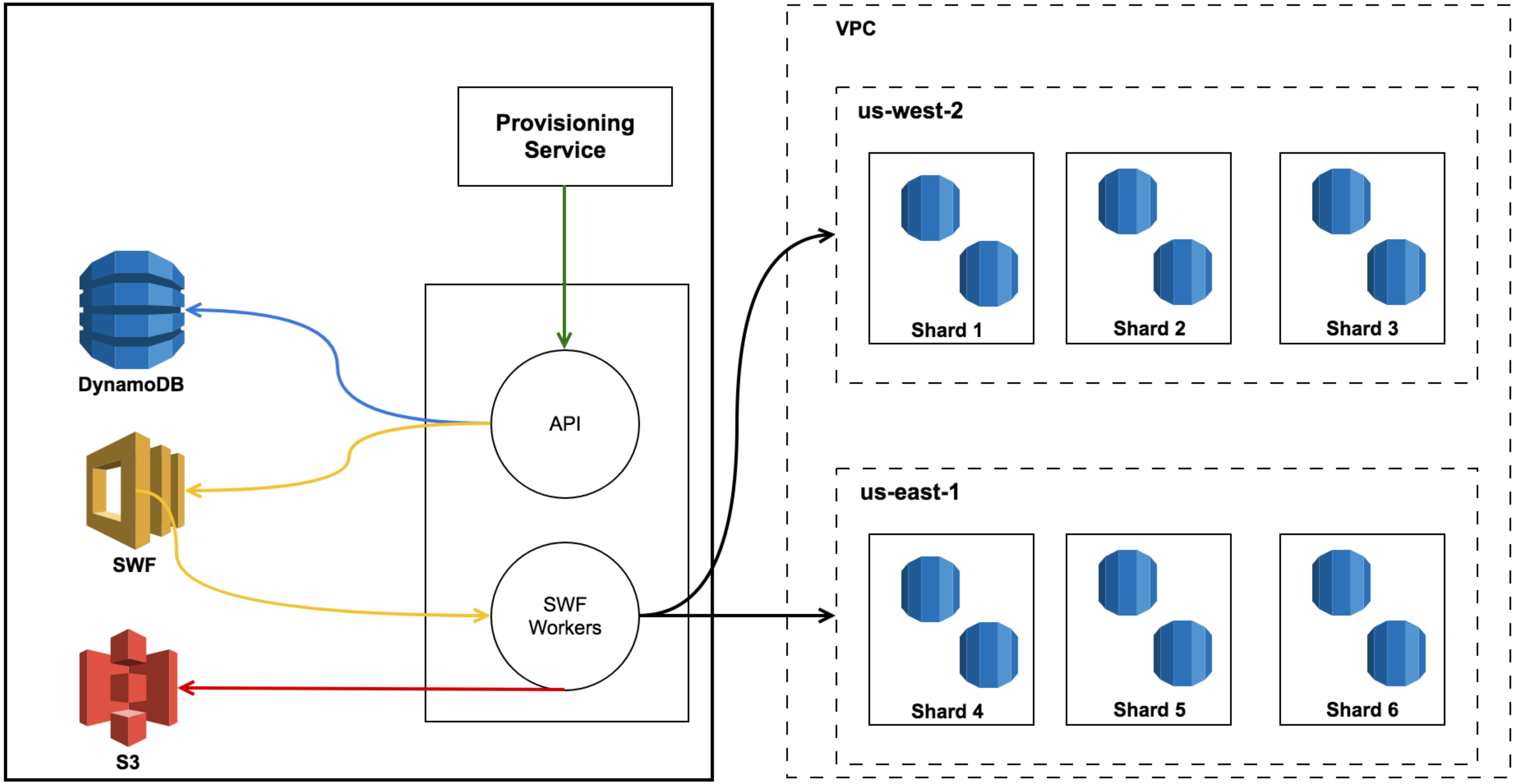
Each tenant gets their own database

Seperate internal user as well, with access locked down

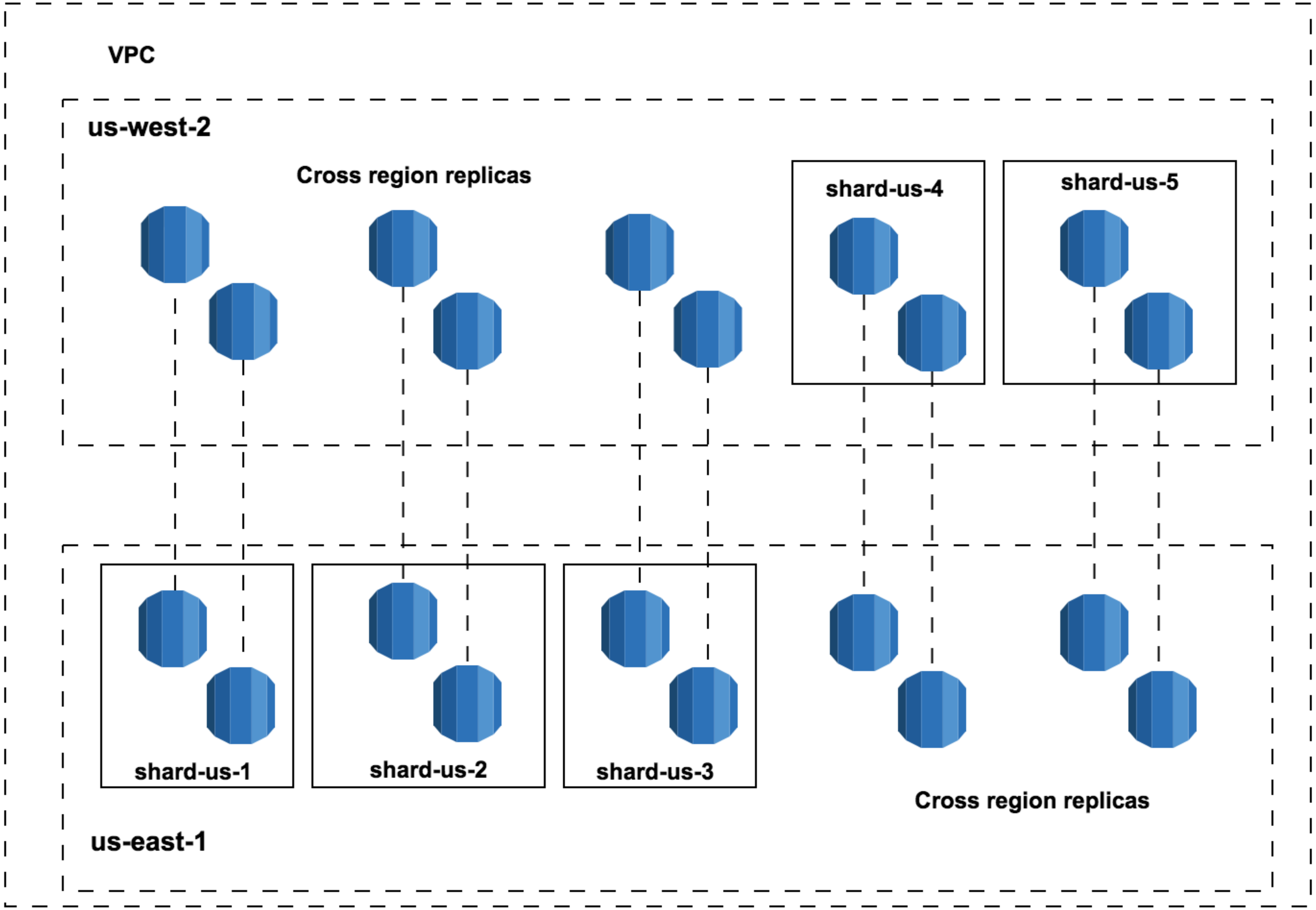
“Unit of operation”, in many ways, is still an RDS instance

Extensively using other AWS services such as CloudFormation, CloudWatch, Performance Insights, etc..

HIGH LEVEL ARCHITECTURE



HIGH AVAILABILITY ARCHITECTURE



Some numbers

~600	>300K	~70TB
RDS Instances	Databases	Data

Operations and Maintenance

Patching and upgrade

Our biggest pain point. Our scale does not allow pg_upgrade based upgrades.

Noisy neighbours

Built tools to move databases between instances

Monitoring

Always look at aggregate level and drill down if needed.

Backups

Both manual and automatic to satisfy our RTO and RPO objectives.

Why RDS, not EC2+PostgreSQL

Architecture

EC2+PG is more suited for a Silo architecture

Not cloud native

Duplicating everything a cloud provider does

High cost

Lot more expensive to run and maintain

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Heavily Used

TSearch

Managing Query Execution

Database Migrations

Log and Query Analysis

HELPS

Consistent Performance

Works well with a wide range of instance sizes

Very little overhead in managing large number of databases

HINDERS

Patching and Upgrading

Replica creation time

Large amount of Writes

No Multi Master Replication

Database creation time

Agenda

Things to consider

Different ways to implement

What we did

How PostgreSQL helps (and doesn’t)

Questions ?

Thank you!

ARUL SHAJI | PRINCIPAL DATABASE CONSULTANT | ATLASSIAN