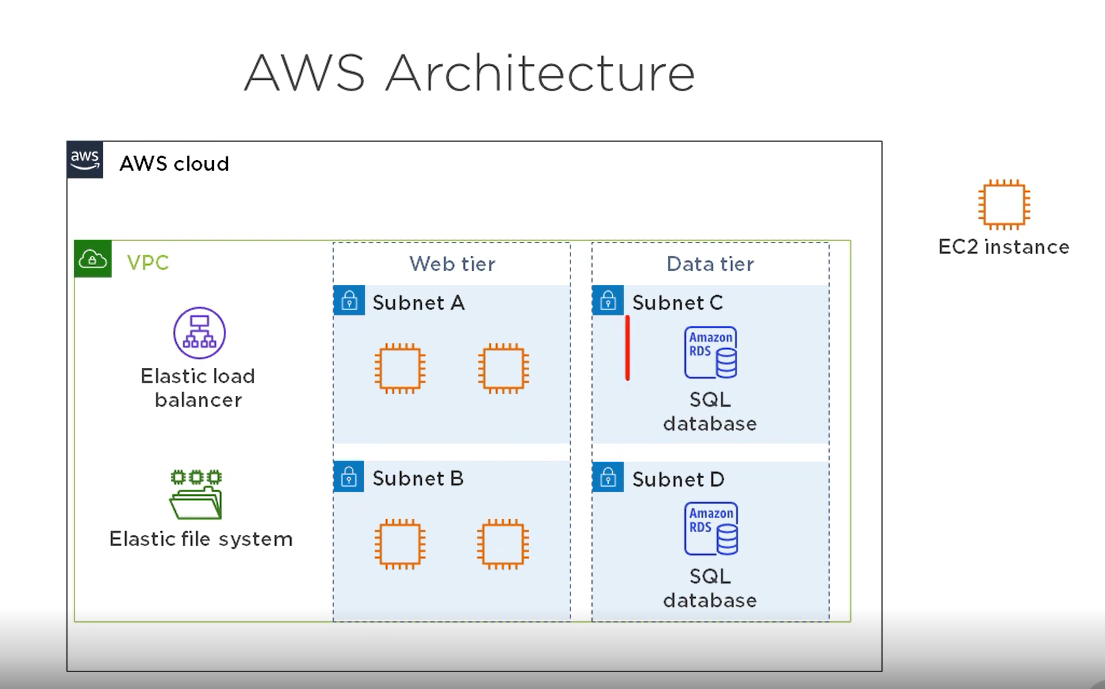
**Architecting for reliability on AWS**

Availability and annual downtime

* 99% - 3 days 15hrs
* 99.9% - 5hrs 45 mins
* 99.99% - about 1 hr
* 99.999% - 5 Mins

**Architecting for 99.9% availability**

Traditional app : Runs on win or linux, you can lift an shift it to aws without changing its code (eg: sql backed web application)



**Calculating Availability - EC2**

The service level agreement (SLA) for each service includes it annual availability. Availability for an eC2 Inst is 90% and failure rate of each inst is 10%

* Calculating avail-redundant instances
* multiply failure rate of 4 inst - 10% \* 10% \* 10% \* 10% = 0.01%
* Subtract product from 100% - 100% - 0.01% = 99.99%

**ELB -** will distribute traffic to the inst while continually monitoring them if 1 inst is down it will stop sending traffic to it.

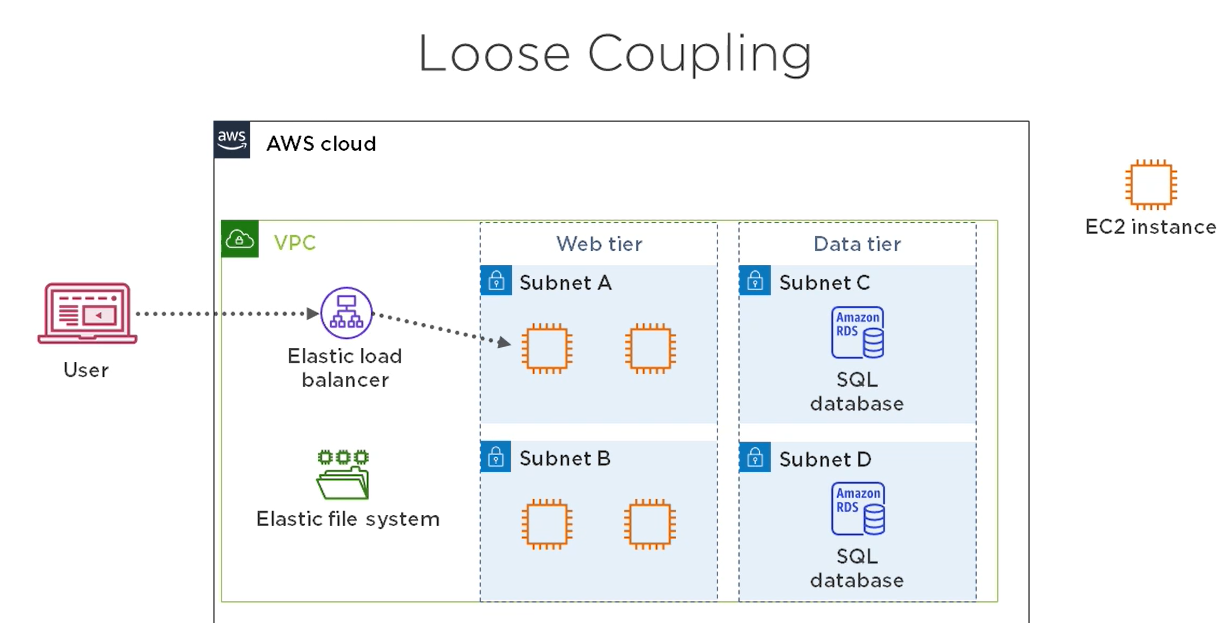
* Calculate avail - ELB
* ELB availability is 99.99%
* Multiply this by the availability of instances - 99.99% \* 99.99% = 99.998%

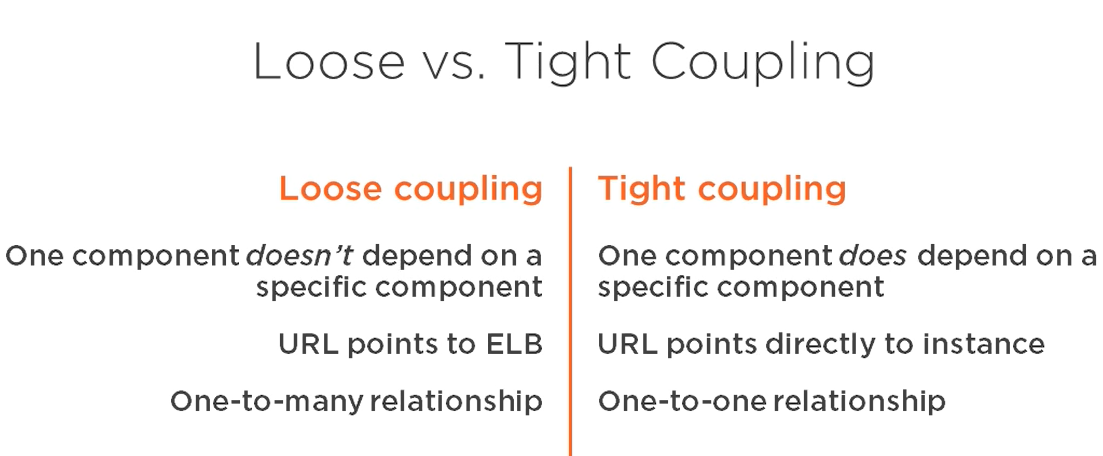
**EFS - Elastic file system**

* EFS avail is 99.99%
* 99.998 \* 99.99% = 99.97%

RDS - Relational Db service

* RDS multi-AZ availability is 99.95%
* 99.95% \* 99.97% = 99.92%

**Loose Coupling**   




**EFS -** Elastic services are always composed of redundant components. Elastic services are always loosely coupled with other resources

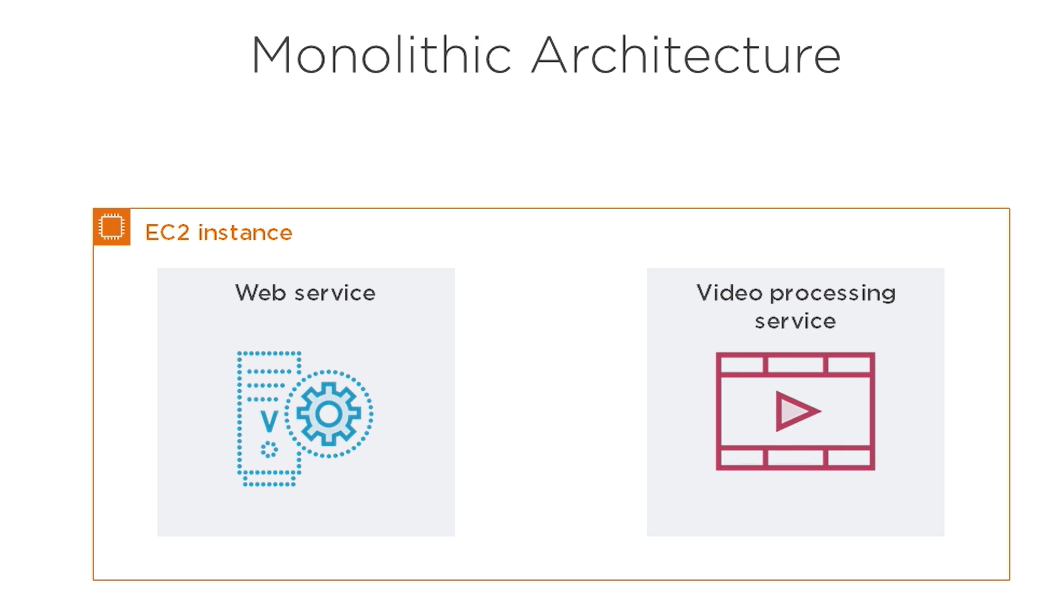
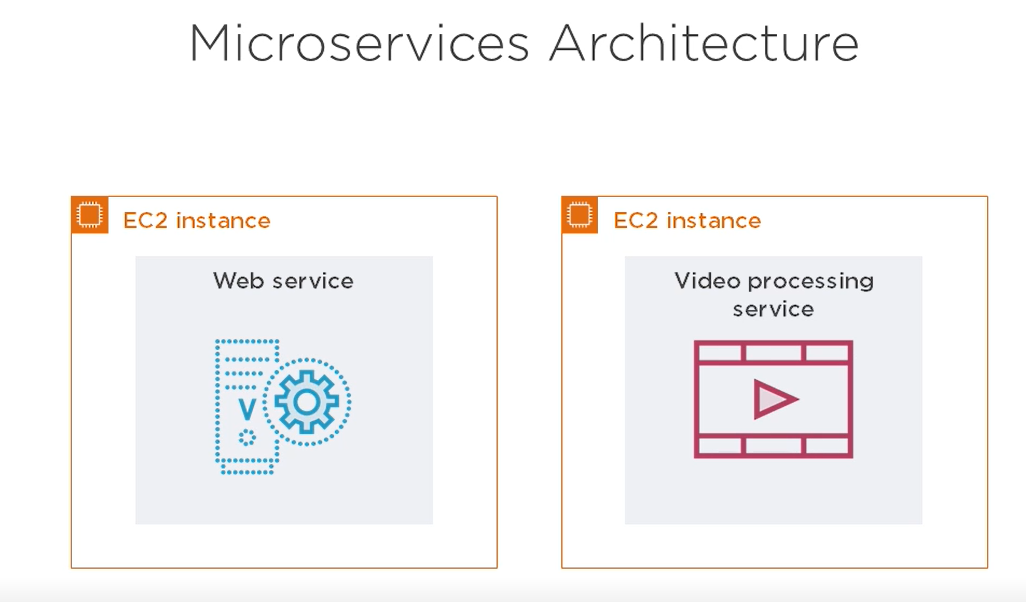
**Redundancy and loose coupling**

* Redundancy --> loose coupling --> availability

**loose coupling and performance :**

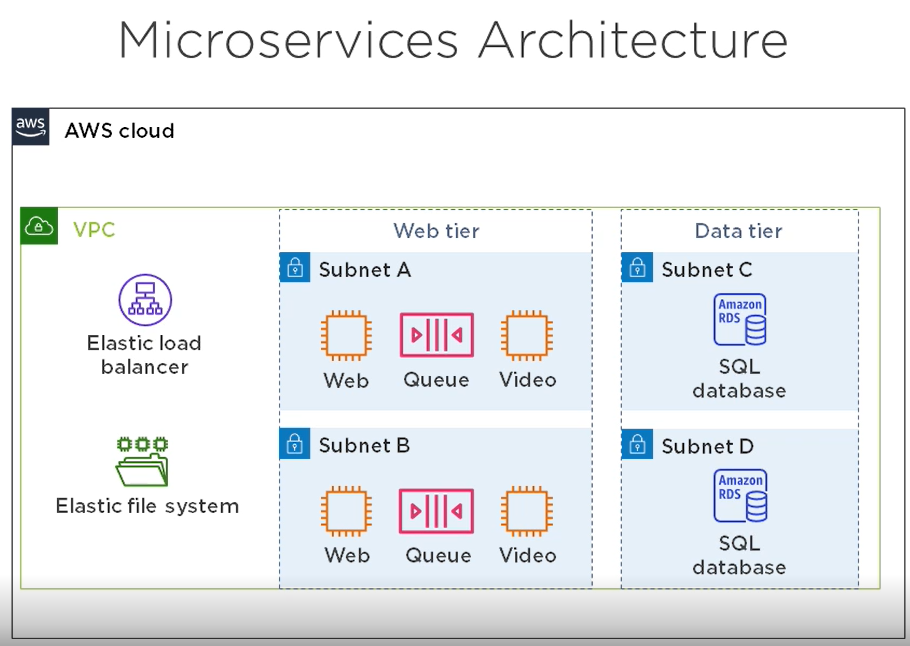
* Independently scale resources without impacting availability
* performance and availability are linked
* poor perform leads to low availability
* good perform helps maintain availability

**Simple Queue Service**

**** ****

**Benefits of micro services** - Easier updates, availability, scalability

**messaging queue using SQS**



**Containers**

* Behave like virtual machines
* Let you run multiple microservices on a single instance
* Processes running inside the container are isolated from the host

**Benefits of containers** - Ease of deployment ,Build an image, deploy it to instances, launch containers from it, Availability

**Docker**

* Image contains everything the service needs
* Launch containers from an image
* Like launching EC2 instances from an AMI

Elastic Container service : ECS clusters contain EC2 instances, can store containers in an elastic container registry(ECR)

**Cloud native applications**

* Depend on a cloud service that can't be deployed on-permises(Eg: SQS, S3, DynamoDB)
* Services instead of servers
* Availability not necessarily higher than traditional applications

**Lambda :**   
Managed serverless compute service,

* The was lambda works is you create a lambda function with you app code, tell lambda the language you app is written in. and simply call the function.
* Lambda supports many lang(C#, java, python, go, PowerShell)
* Availability is 99.95%

**S3** : unlimited file storage, static web hosting, avail is 99.9%

**DynamoDB**

* Managed, nonrelational database
* Global tables feature replicates tables across multiple regions
* Availability with global tables is 99.999%

**Calculating Availability** : 99.95% \* 99.9% \* 99.999% = 99.85%