Errors or corrections? Email us at aws-course-feedback@amazon.com.
Other questions? Email us at aws-training-info@amazon.com.

Architecting with AWS

Architecting in the Cloud

Architecting in the Cloud | What we'll cover

1. Five benefits of the cloud
2. Seven best practices for building systems with AWS

1

Five benefits of the cloud

What makes the cloud attractive?

- **Abstract Resources**
  - Focus on your needs, not hardware specs. As needs change, so should your resources.
- **On-Demand Provisioning**
  - Ask for what you need, exactly when you need it; get rid of it when you don't.
- **Scalability in Minutes**
  - Scale out or in, up or down, depending on usage or needs

What makes the cloud attractive?

- **Pay Per Consumption**
  - No long-term commitments. Pay only for what you use.
- **Efficiency of Experts**
  - Utilize the skills, knowledge and resources of experts.

Architecting in the Cloud | Seven best practices for building systems with AWS

2

Seven best practices for building systems with AWS

There are 7 best practices to remember…

1. Design for failure and nothing fails

2. Loose coupling sets you free

3. Implement elasticity

4. Build security in every layer

5. Don’t fear constraints

6. Think parallel

7. Leverage different storage options

"Everything fails, all the time."

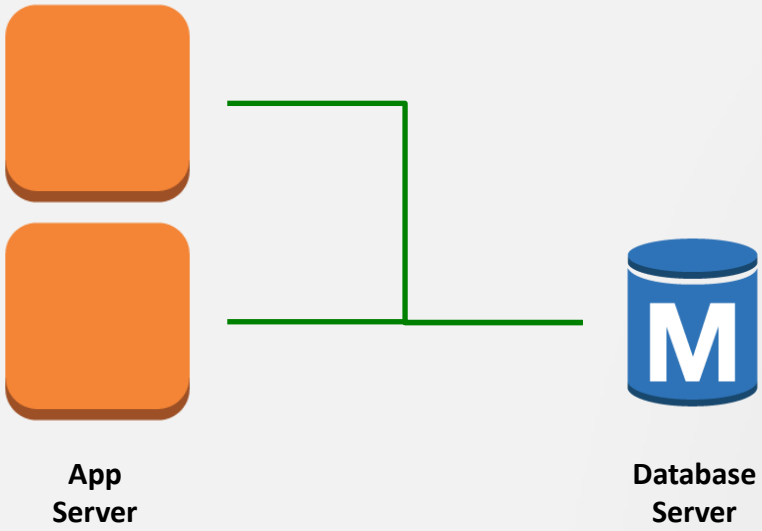
Werner Vogels, CTO, Amazon.com

Design for failure

- Avoid single points of failure

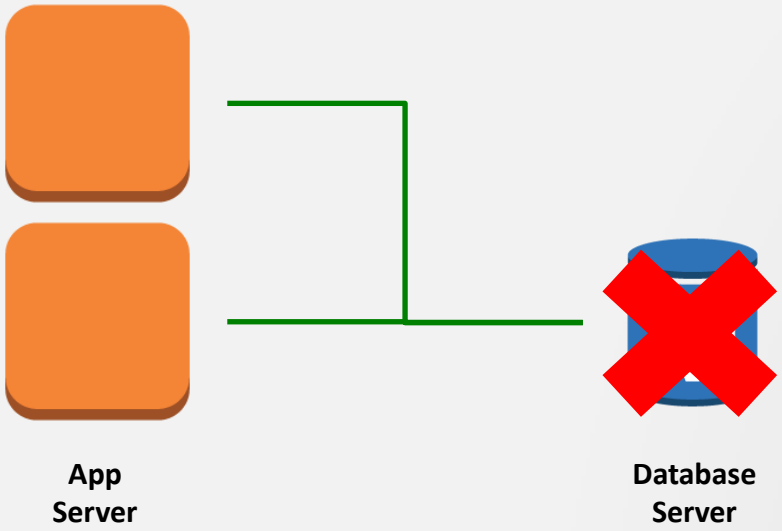
Design for failure

- Avoid single points of failure



Design for failure

- Avoid single points of failure

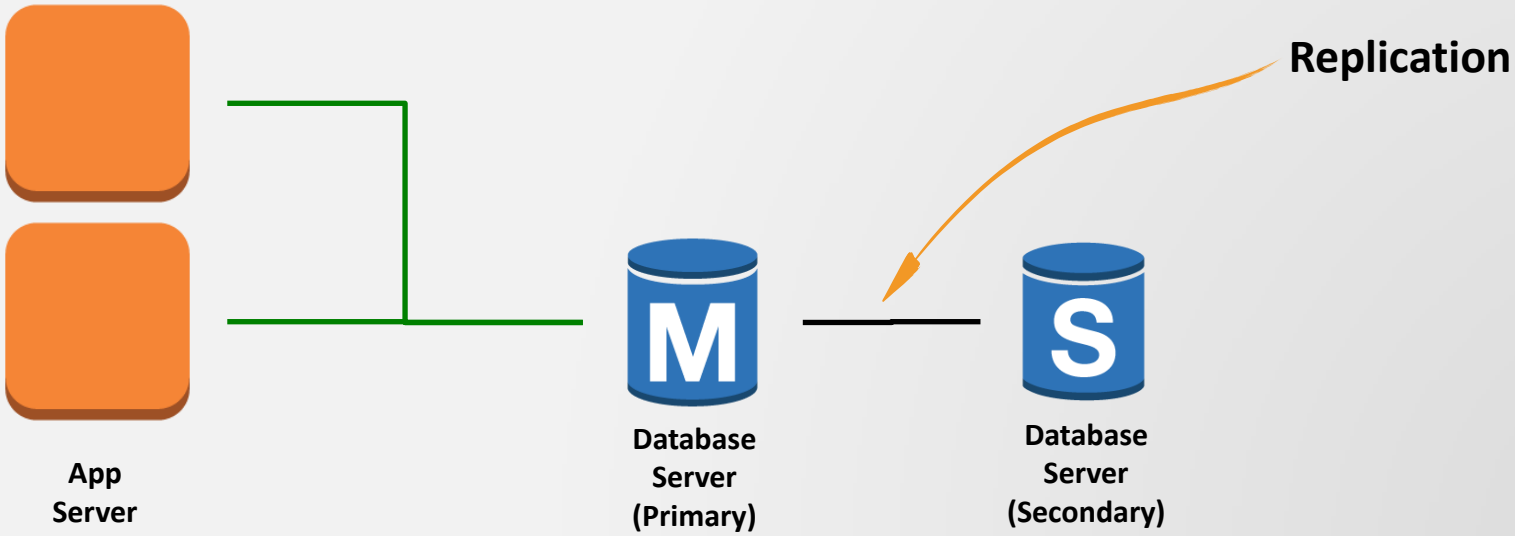


Design for failure

- Avoid single points of failure
- Assume everything fails and design backwards
  - **Goal: Applications should continue to function even if the underlying physical hardware fails or is removed/replaced.**

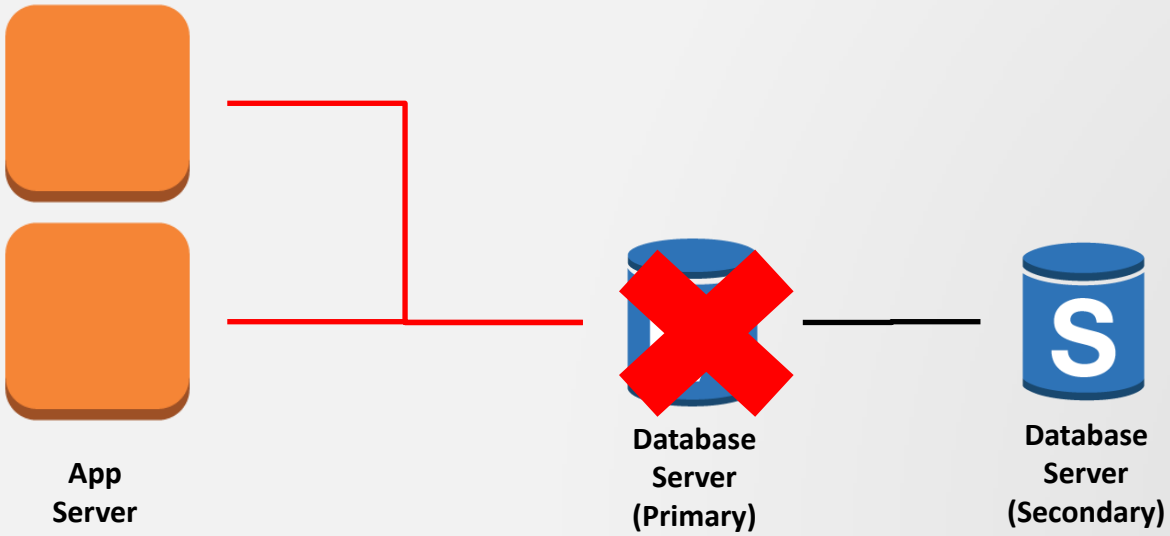
Design for failure

- Avoid single points of failure
- Assume everything fails and design backwards
  - **Goal: Applications should continue to function even if the underlying physical hardware fails or is removed/replaced.**



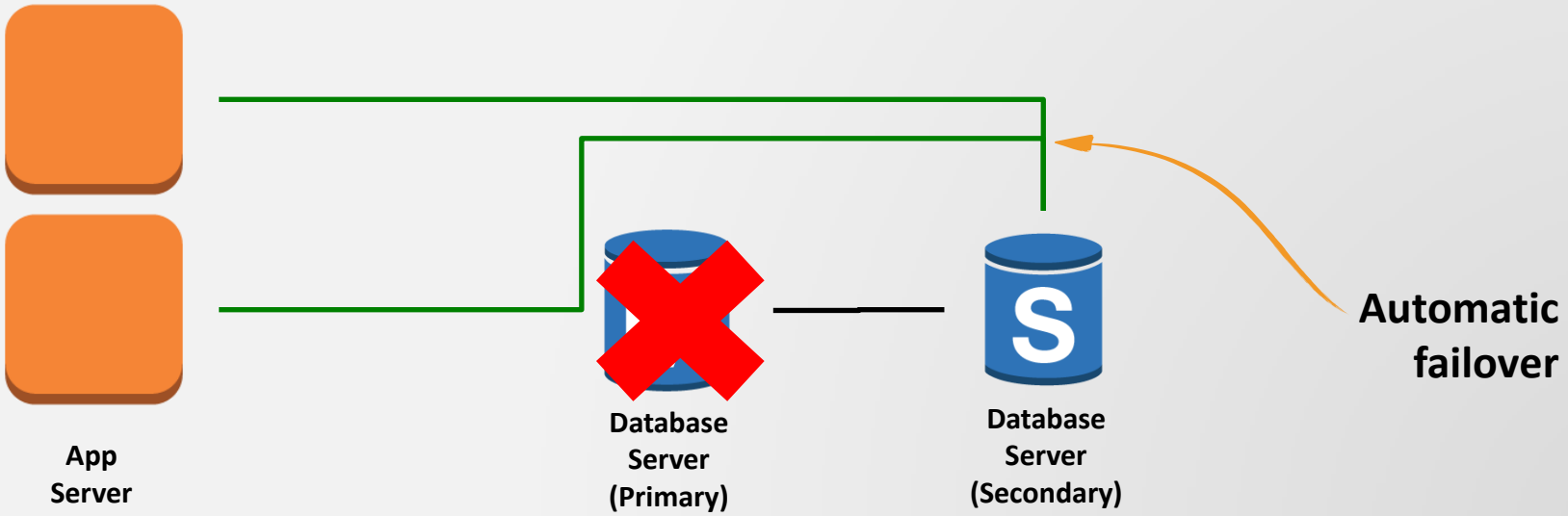
Design for failure

- Avoid single points of failure
- Assume everything fails and design backwards
  - **Goal: Applications should continue to function even if the underlying physical hardware fails or is removed/replaced.**



Design for failure

- Avoid single points of failure
- Assume everything fails and design backwards
  - **Goal: Applications should continue to function even if the underlying physical hardware fails or is removed/replaced.**



Loose coupling sets you free

Loose coupling sets you free

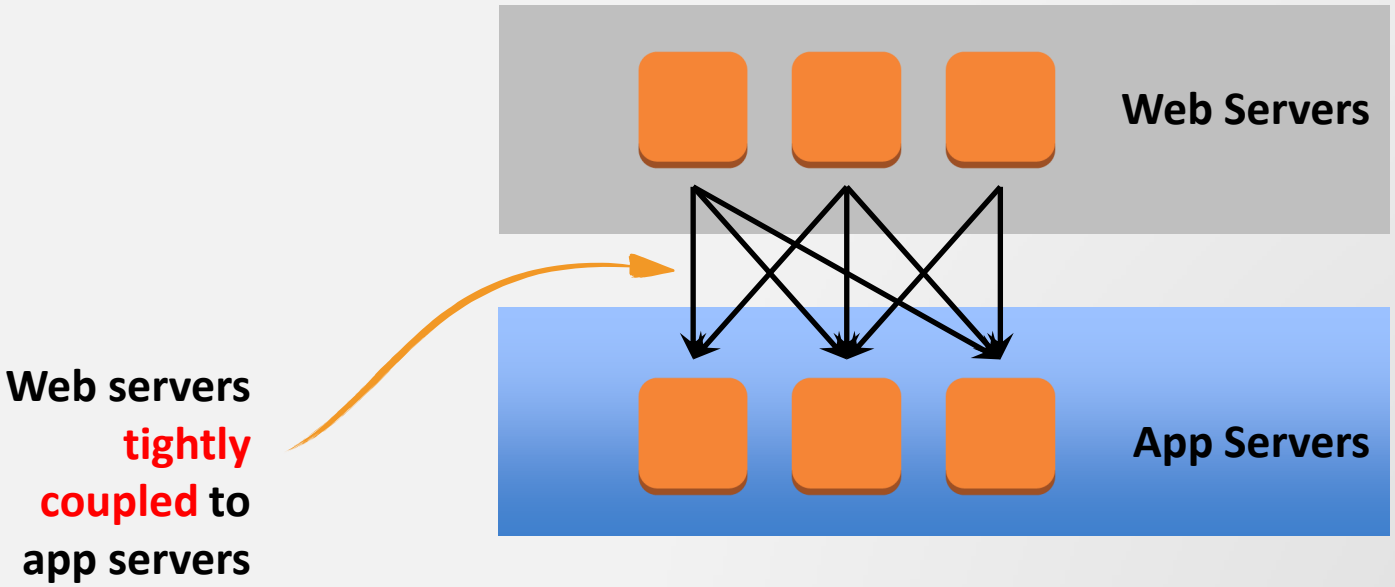
- Design architectures with independent components
  - The more loosely they're coupled, the bigger they scale
- Design every component as a black box

Loose coupling sets you free

- Load balance clusters
- Use a queue to pass messages between components

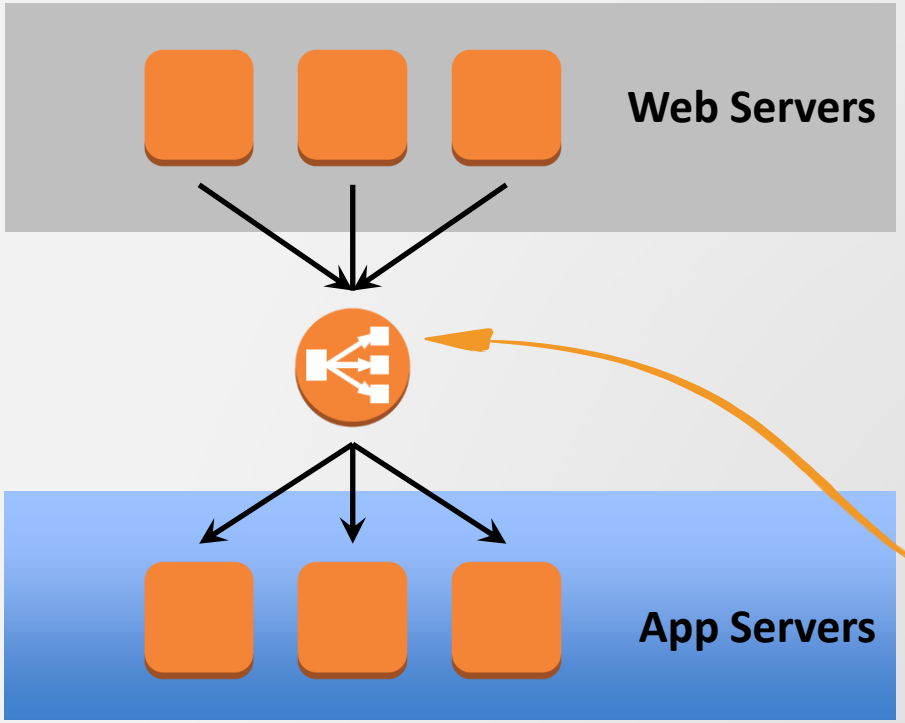
Loose coupling sets you free

- Load balance clusters



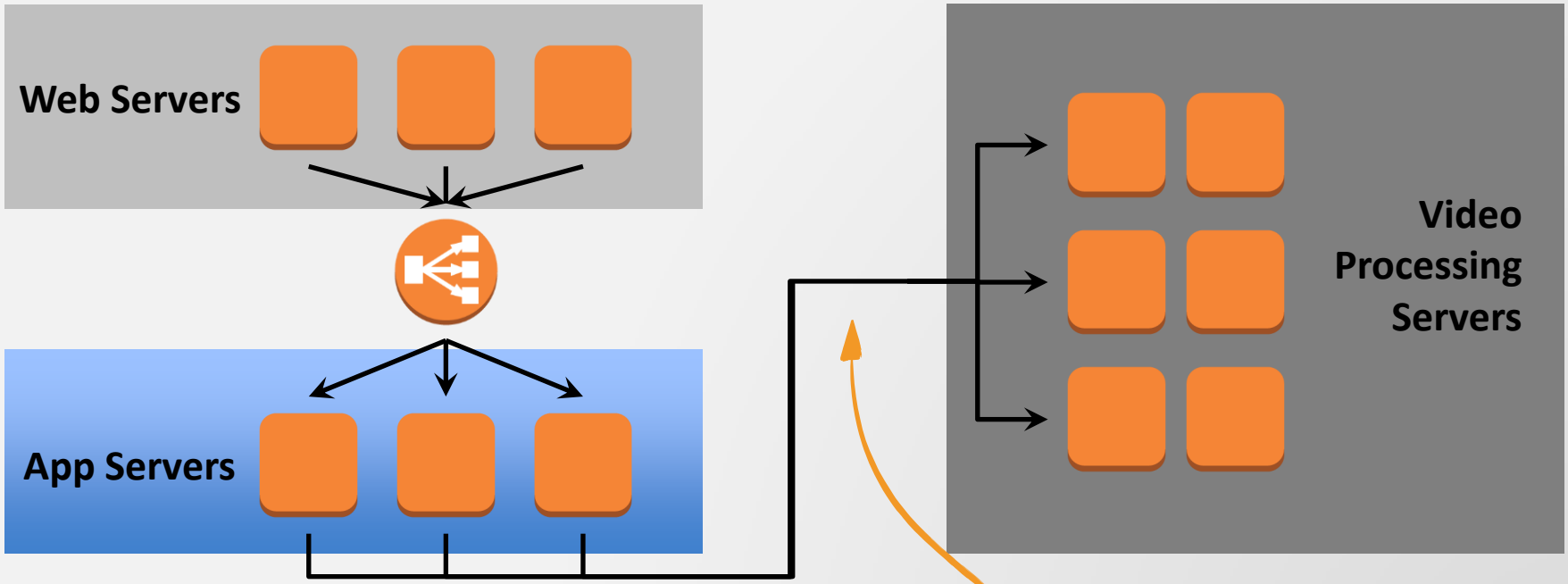
Loose coupling sets you free

- Load balance clusters



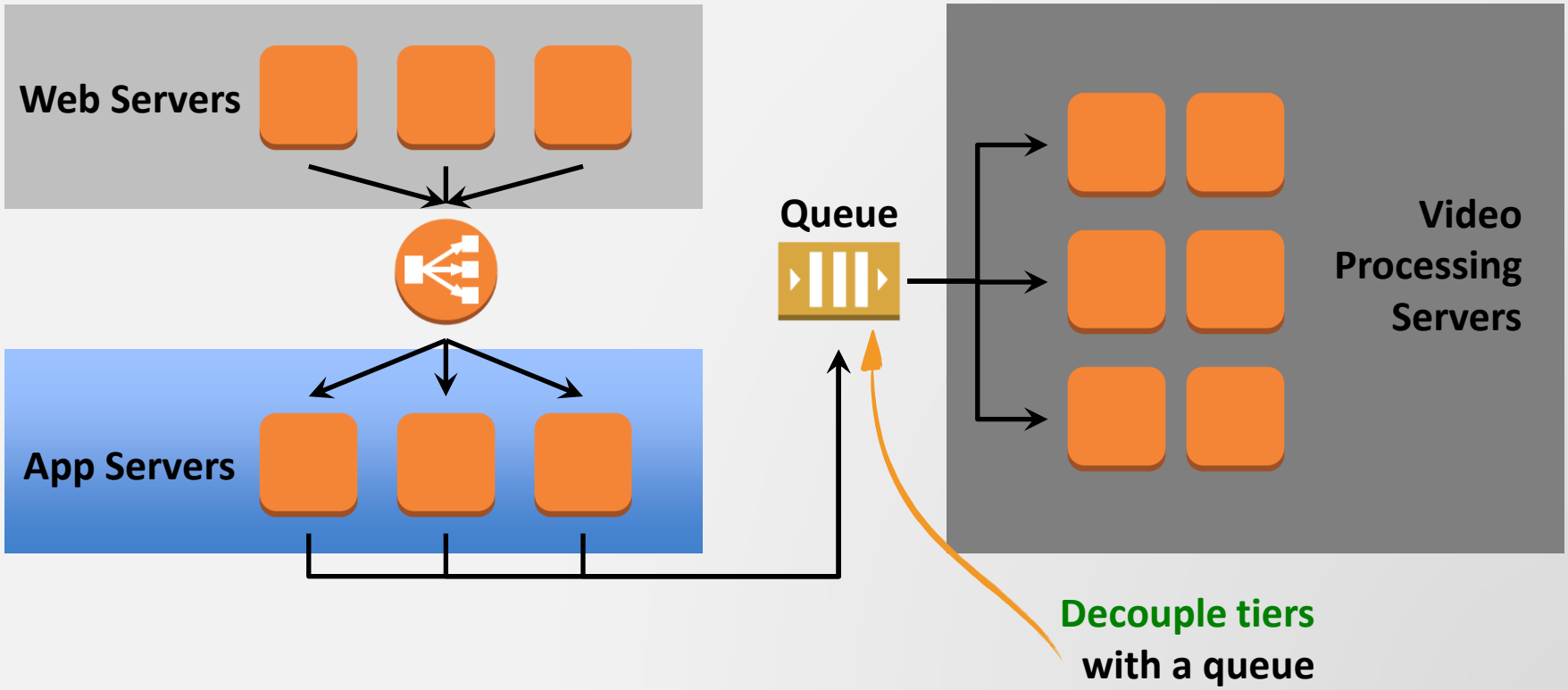
Loose coupling sets you free

- Use a queue to pass messages between components



Loose coupling sets you free

- Use a queue to pass messages between components



Implement elasticity

Implement elasticity

- Elasticity is a fundamental property of the cloud

Implement elasticity

- Elasticity is a fundamental property of the cloud
- Don’t assume the health, availability, or fixed location of components

Implement elasticity

- Elasticity is a fundamental property of the cloud
- Don’t assume the health, availability, or fixed location of components
- Use designs that are resilient to reboot and re-launch

Implement elasticity

- Elasticity is a fundamental property of the cloud
- Don’t assume the health, availability, or fixed location of components
- Use designs that are resilient to reboot and re-launch
- Bootstrap your instances
  - When an instance launches, it should ask “**Who am I and what is my role?**”

Implement elasticity

- Elasticity is a fundamental property of the cloud
- Don’t assume the health, availability, or fixed location of components
- Use designs that are resilient to reboot and re-launch
- Bootstrap your instances
  - When an instance launches, it should ask “Who am I and what is my role?”
- Favor dynamic configuration

Build security in every layer

Security is a shared responsibility. You decide how to:

Build security in every layer

Security is a shared responsibility. You decide how to:

- Encrypt data in transit and at rest
- Enforce principle of least privilege
- Create distinct, restricted Security Groups for each application role
  - Restrict external access via these security groups
- Use multi-factor authentication

Don’t fear constraints

Don’t fear constraints

- Need more RAM?
  - Consider distributing load across machines or a shared cache

Don’t fear constraints

- Need more RAM?
  - Consider distributing load across machines or a shared cache
- Need better IOPS for database?
  - Instead, consider multiple read replicas, sharding, or DB clustering

Don’t fear constraints

- Need more RAM?
  - Consider distributing load across machines or a shared cache
- Need better IOPS for database?
  - Instead, consider multiple read replicas, sharding, or DB clustering
- Hardware failed or config got corrupted?
  - “Rip and replace”—Simply toss bad instances and instantiate replacement

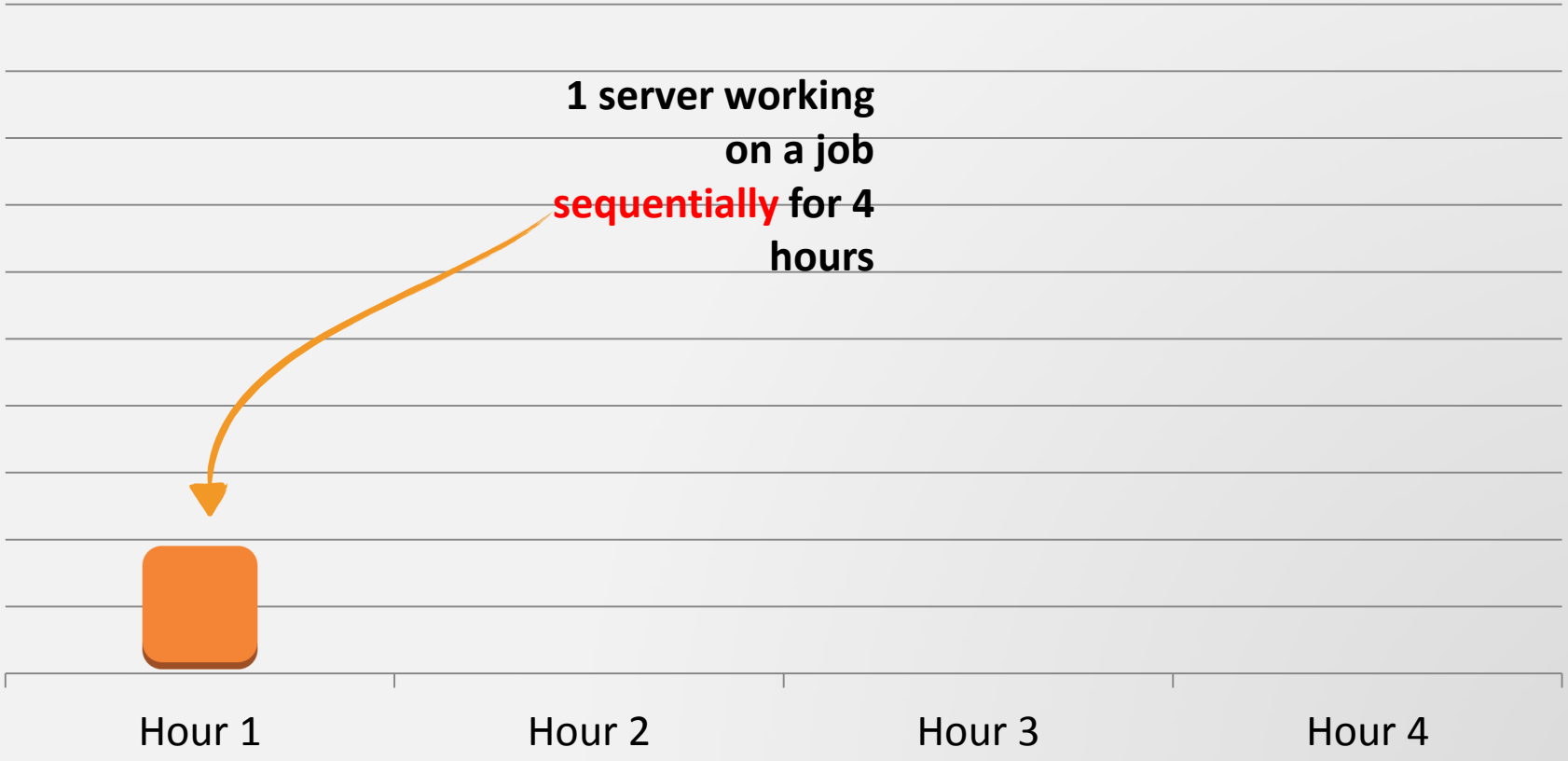
Think parallel

Think parallel

- Experiment with parallel architectures

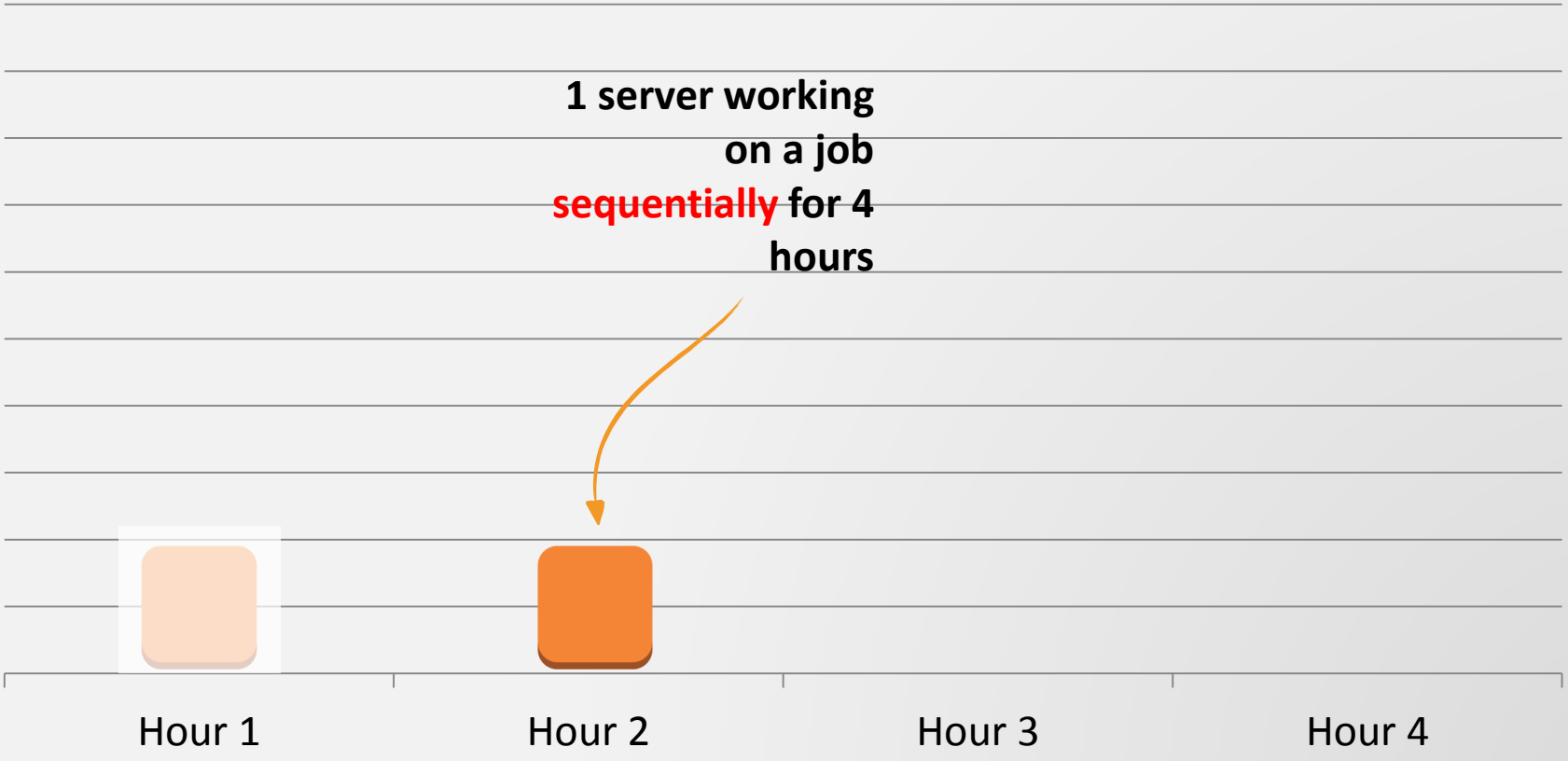
Think parallel

- Experiment with parallel architectures



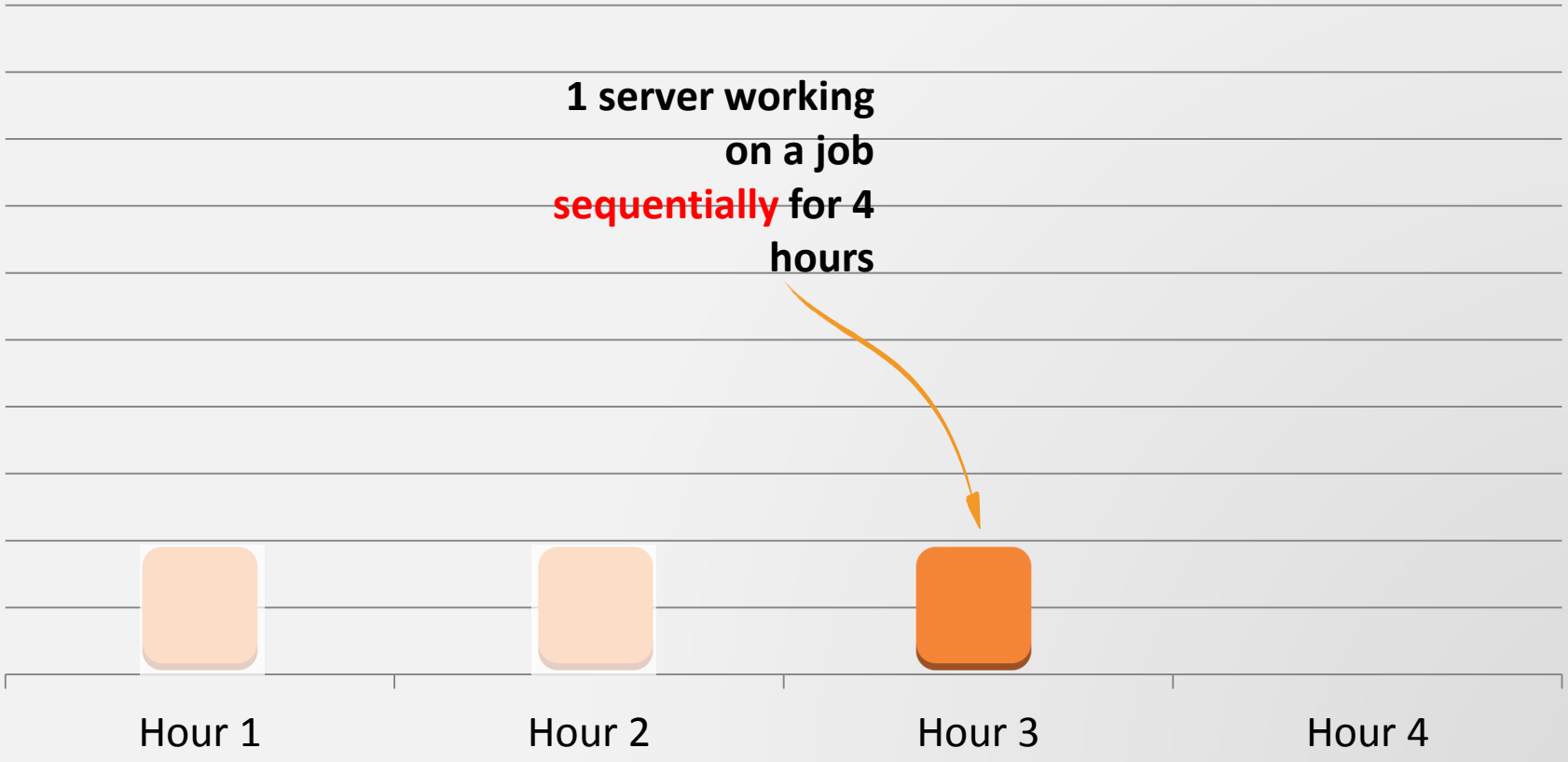
Think parallel

- Experiment with parallel architectures



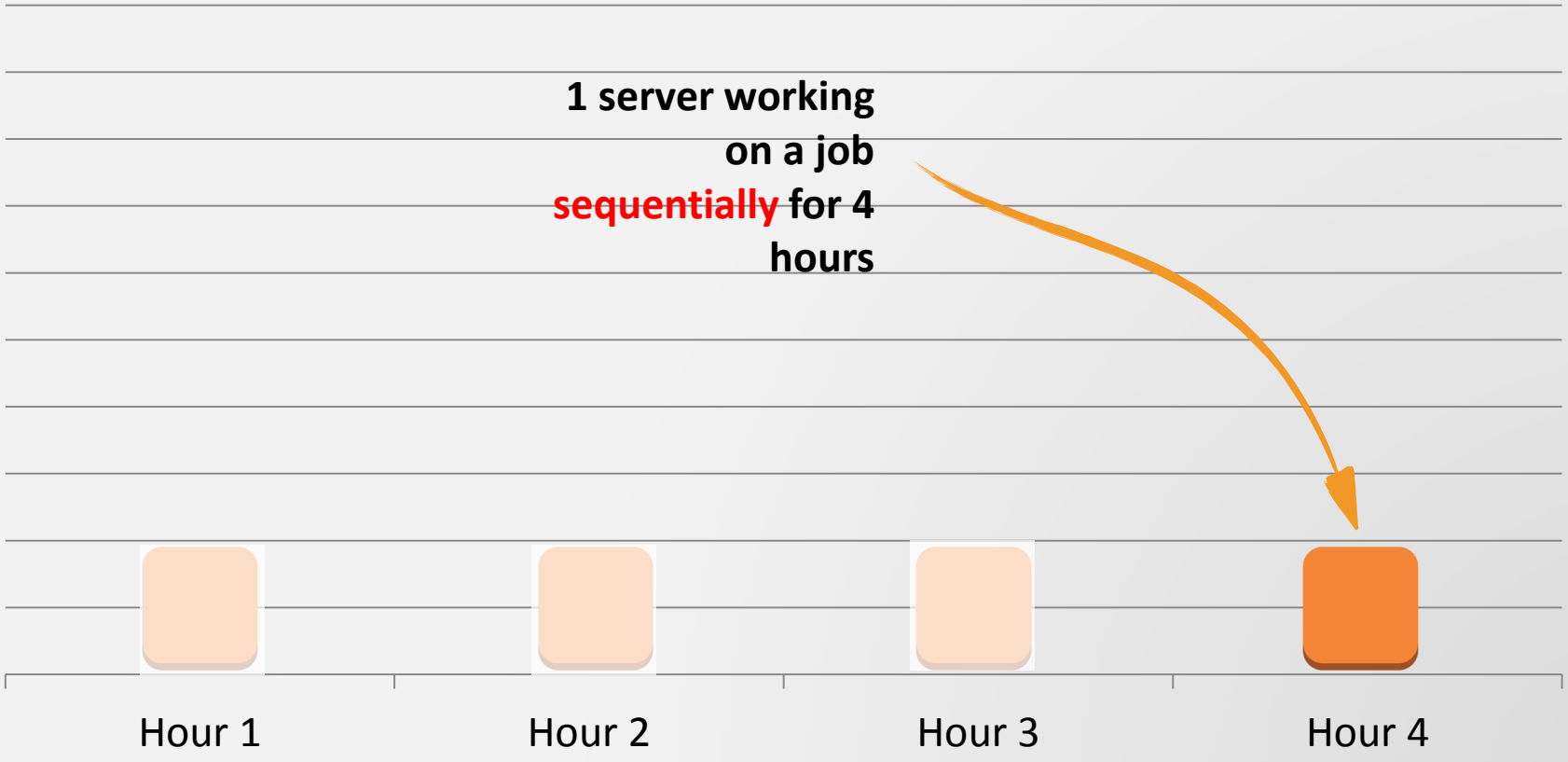
Think parallel

- Experiment with parallel architectures



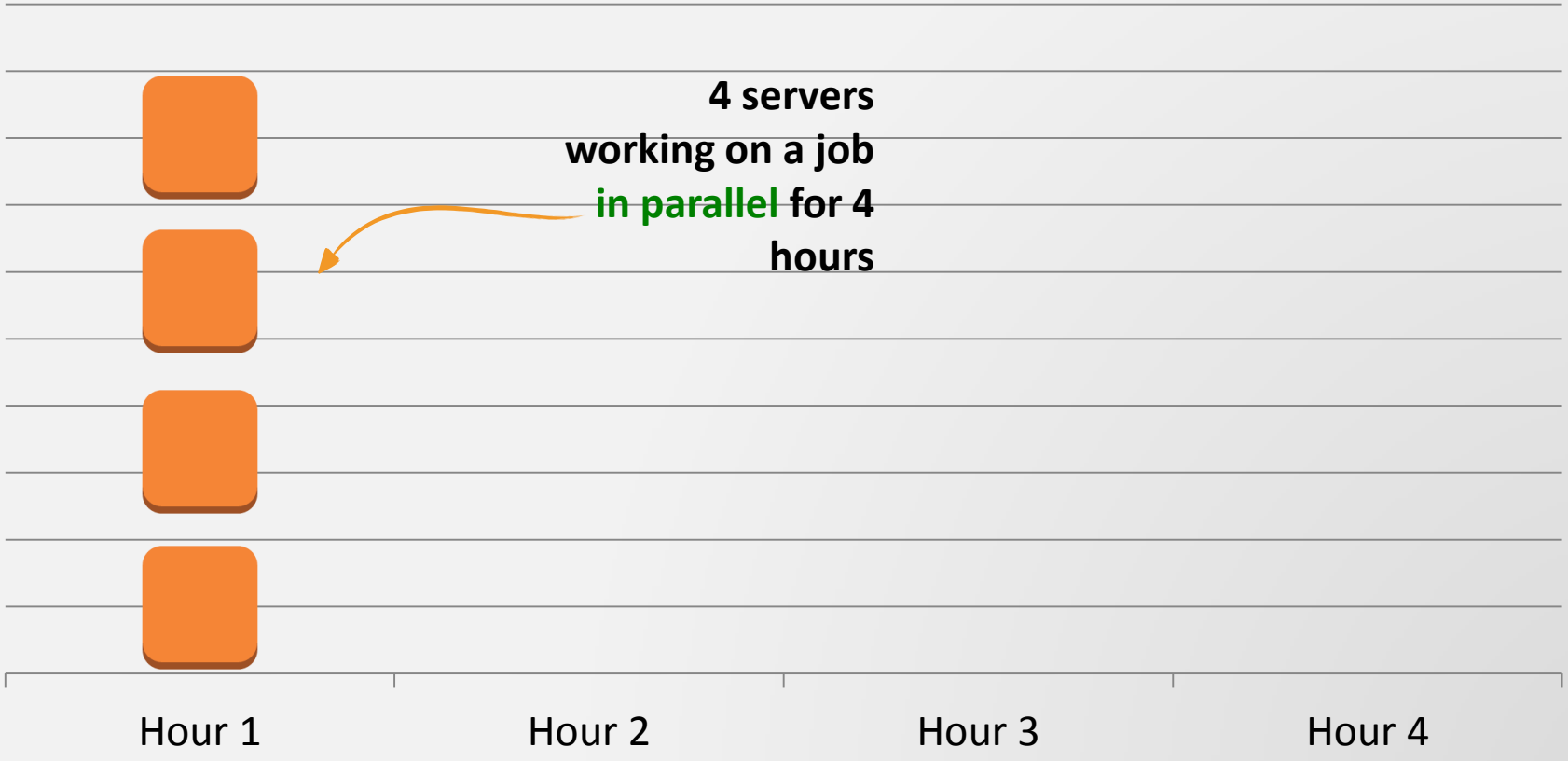
Think parallel

- Experiment with parallel architectures



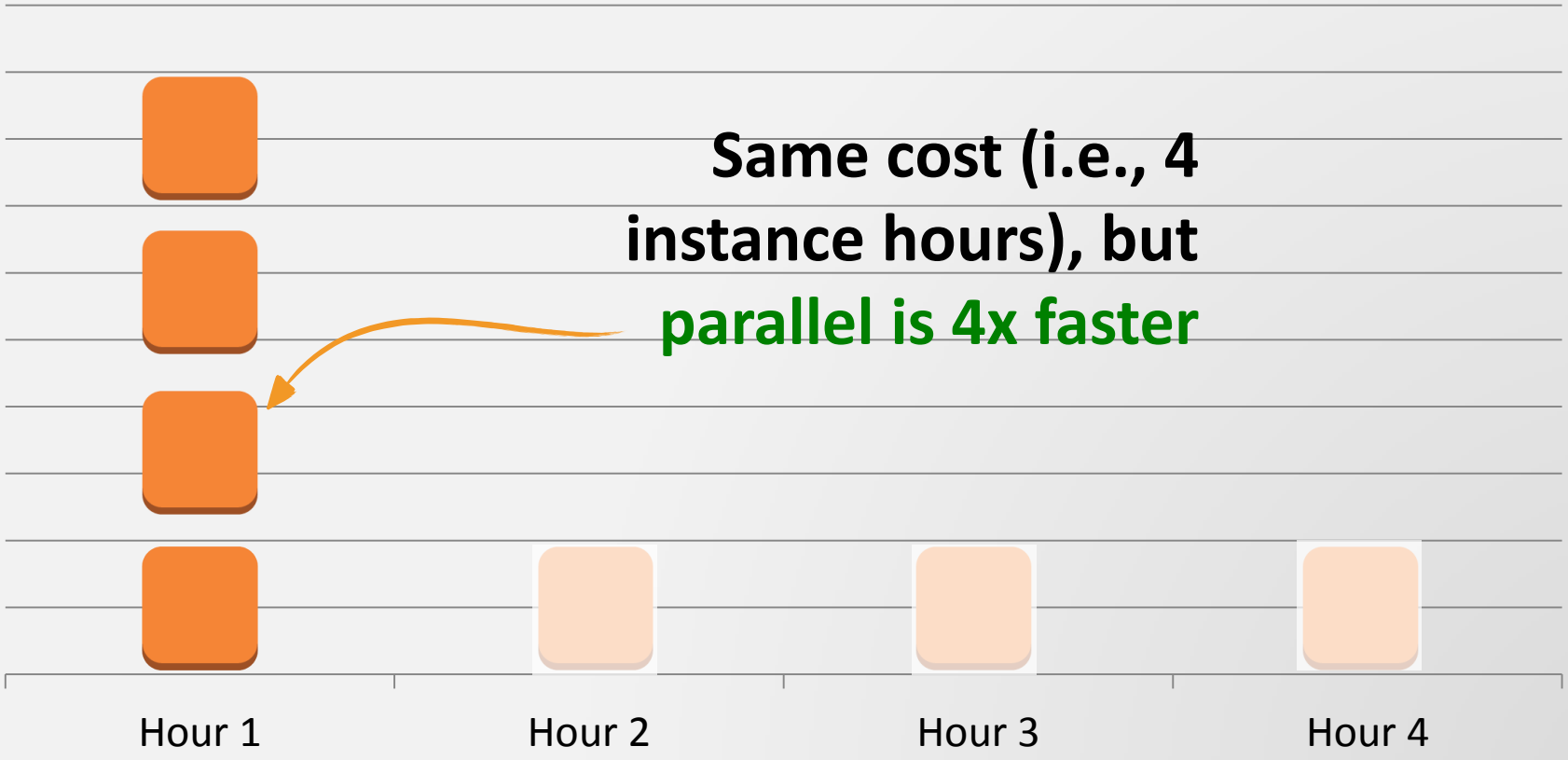
Think parallel

- Experiment with parallel architectures



Think parallel

- Experiment with parallel architectures



Leverage many storage options

Leverage many storage options

One size does not fit all

- Object storage
- Content delivery network/edge caching
- Block storage
- Relational database
- NoSQL

Let's review:

- List five benefits cloud services offer
- List the seven AWS best practices