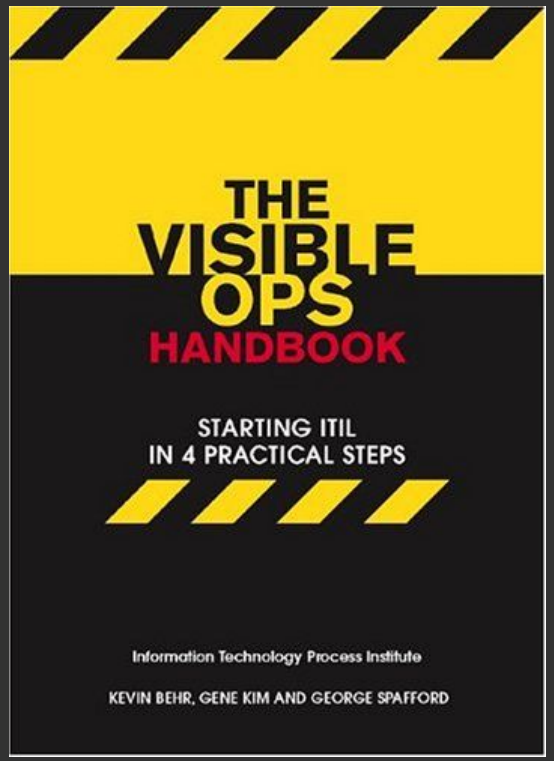

Infraestrutura Imutável
@fernandoike

“Sinônimos”

- Immutable Infrastructure
- Immutable Server
- Immutable Delivery
- Golden Images
- Phoenix Servers vs Snowflake Servers
- Pets vs Cattle
- Infrastructure as Code



Netflix Technology Blog Follow
Learn more about how Netflix designs, builds, and operates our systems and engineering organizations
Aug 13, 2011 · 3 min read

Building with Legos

In the six years that I have been involved in building and releasing software here at Netflix, the process has evolved and improved significantly. When I started, we would build a WAR, get it setup and tested on a production host, and then run a script that would stop tomcat on the host being pushed to, rsync the directory structure and then start tomcat again. Each host would be manually pushed to using this process, and even with very few hosts this took quite some time and a lot of human interaction (potential for mistakes).

cloudscaling

Architectures for open and scalable clouds

February 14, 2012

Randy Bias, CTO & Co-founder

CloudConnect

SnowflakeServer

Martin Fowler
10 July 2012

It can be finicky business to keep a production server running. You have to ensure the operating system and any other dependent software is properly patched to keep it up to date. Hosted applications need to be upgraded regularly. Configuration changes are regularly needed to tweak the environment so that it runs efficiently and communicates properly with other systems. This requires some mix of command-line invocations, jumping between GUI screens, and editing text files.

PhoenixServer

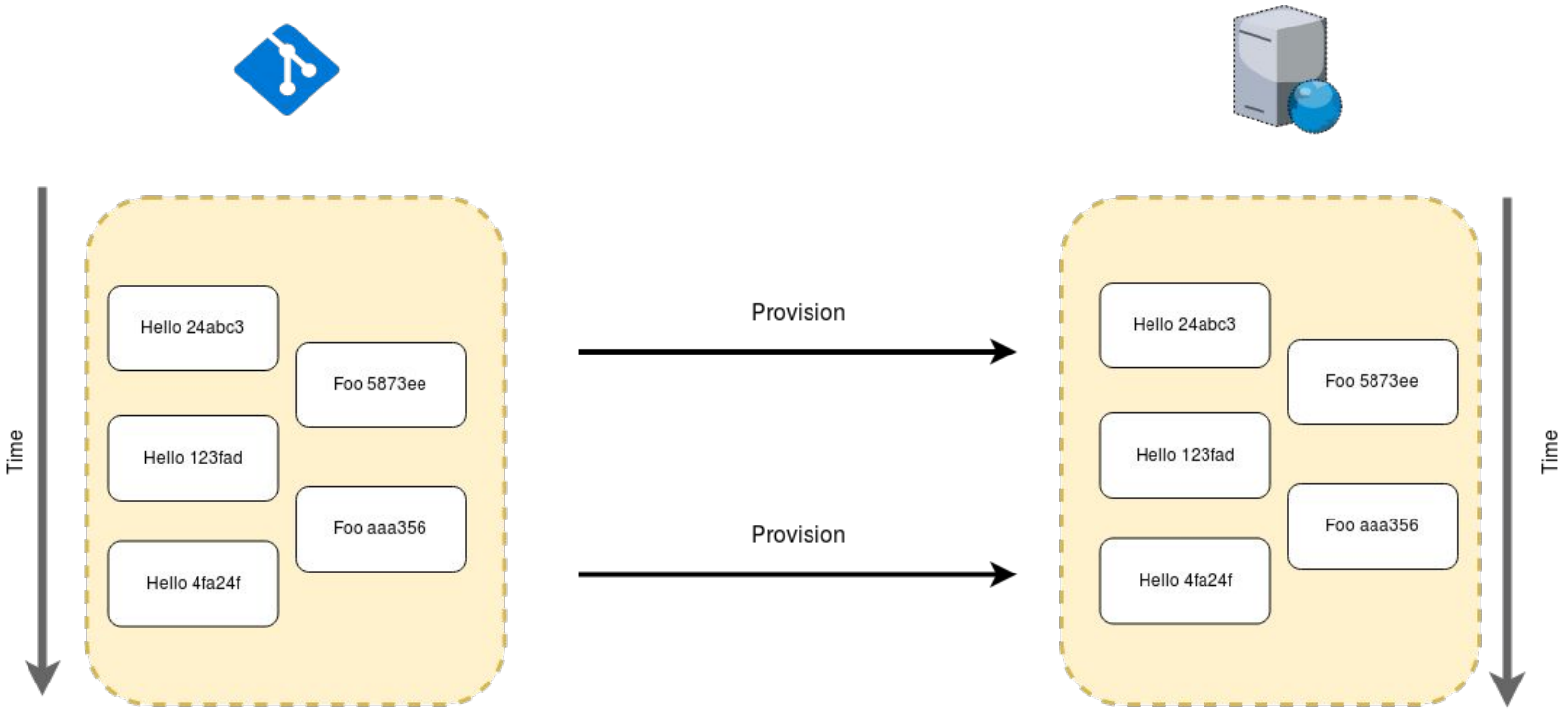
Martin Fowler
10 July 2012

One day I had this fantasy of starting a certification service for operations. The certification assessment would consist of a colleague and I turning up at the corporate data center and setting about critical production servers with a baseball bat, a chainsaw, and a water pistol. The assessment would be based on how long it would take for the operations team to get all the applications up and running again.

Cenário mais comum - Snowflakes

- A construção do artefato é realizado numa ferramenta de Entrega Contínua
- As dependências são (re)instaladas a cada lançamento de versão do artefato
- O artefato é implementado nos servidores de homologação
- O artefato é implementado nos servidores de produção

Snowflake Servers



Uma (falsa) premissa

"A maneira mais econômica de garantir que o comportamento de dois servidores permanecerá completamente idêntico é sempre a implementar as mesmas alterações na mesma ordem em ambos anfitriões."

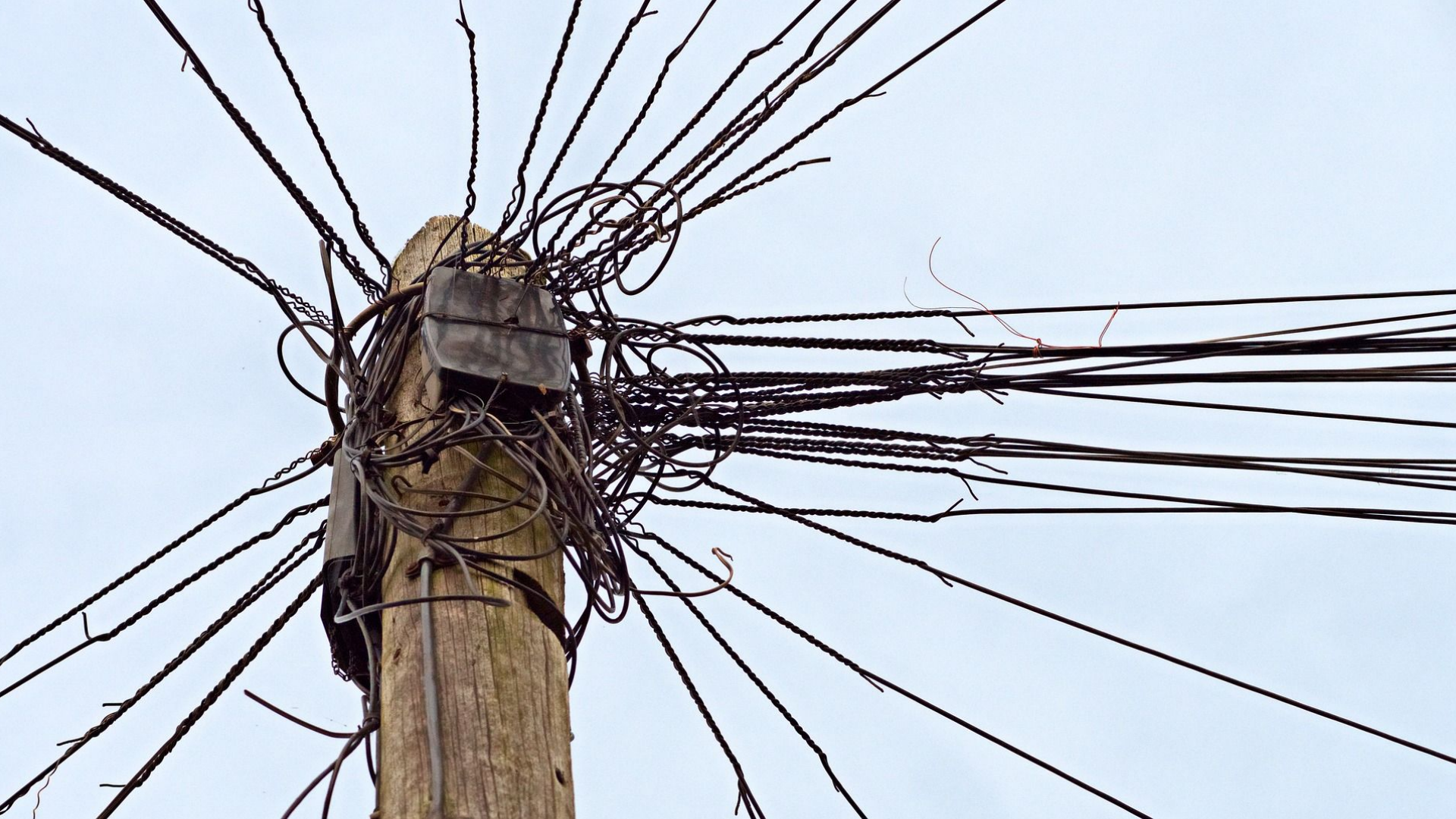
John Willis - @botchagalupe

Por que (Quando) a ordem dos comandos é importante?

- Dependência Circular
- Comandos certos na ordem errada
- Pacotes certos na ordem errada

Efeitos colaterais

- O repositório do sistema de empacotamento (Composer, PIP, GEM, apt, etc.) está indisponível
- As dependências quebram porque a biblioteca foo-1.15-1 não está mais disponível
- A biblioteca de dependência foobar-1.15-2 quebrou a construção do artefato



Problemas comuns em servidores mutáveis

- Aumento da complexidade operacional
  - Mais etapas no Pipeline
  - Maior tempo de Lead Time
  - Mais suscetível a falhas de terceiros, ex: repositórios externos

Dictionary

immutable

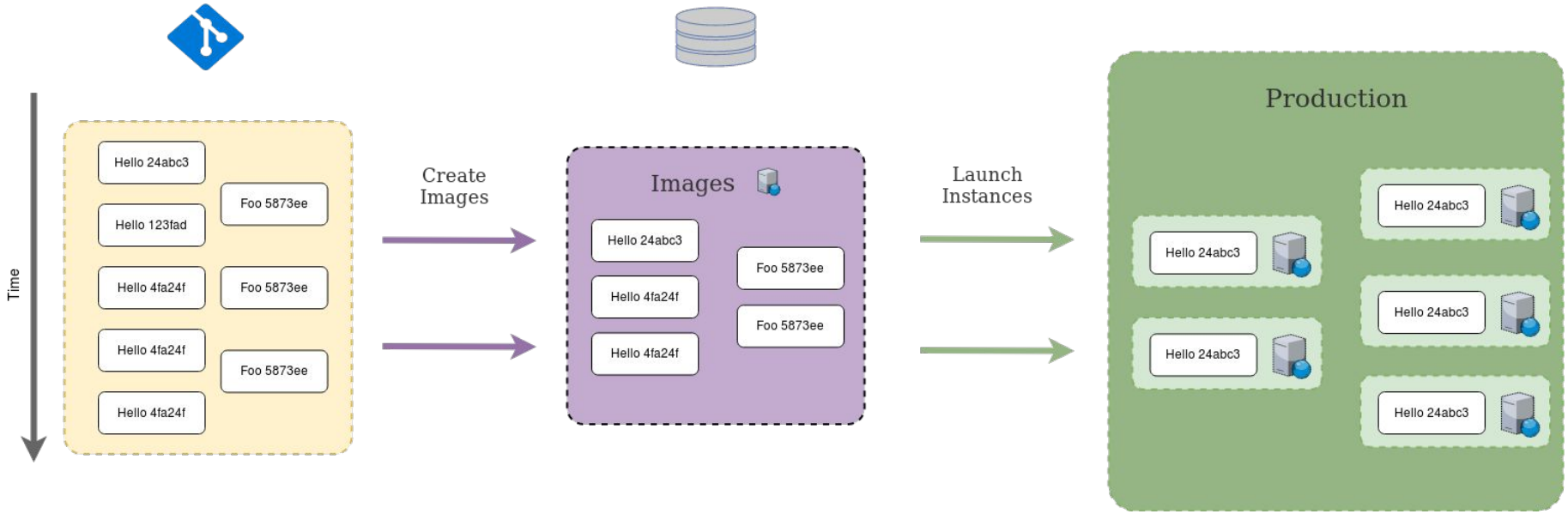
im·mu·ta·ble

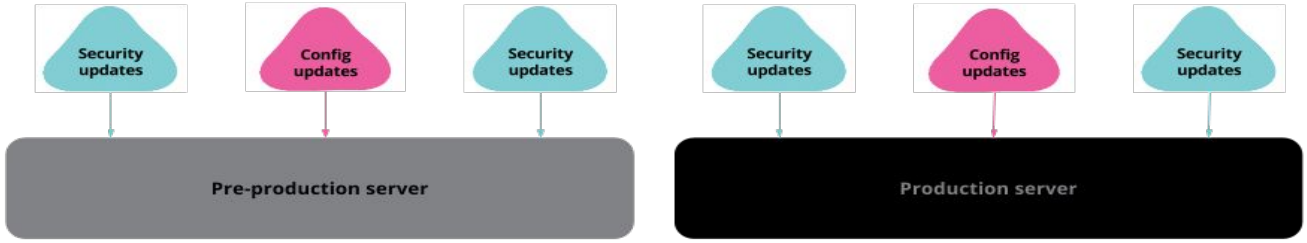
/i(m)ˈmyo͞odəb(ə)l/

adjective
adjective: **immutable**

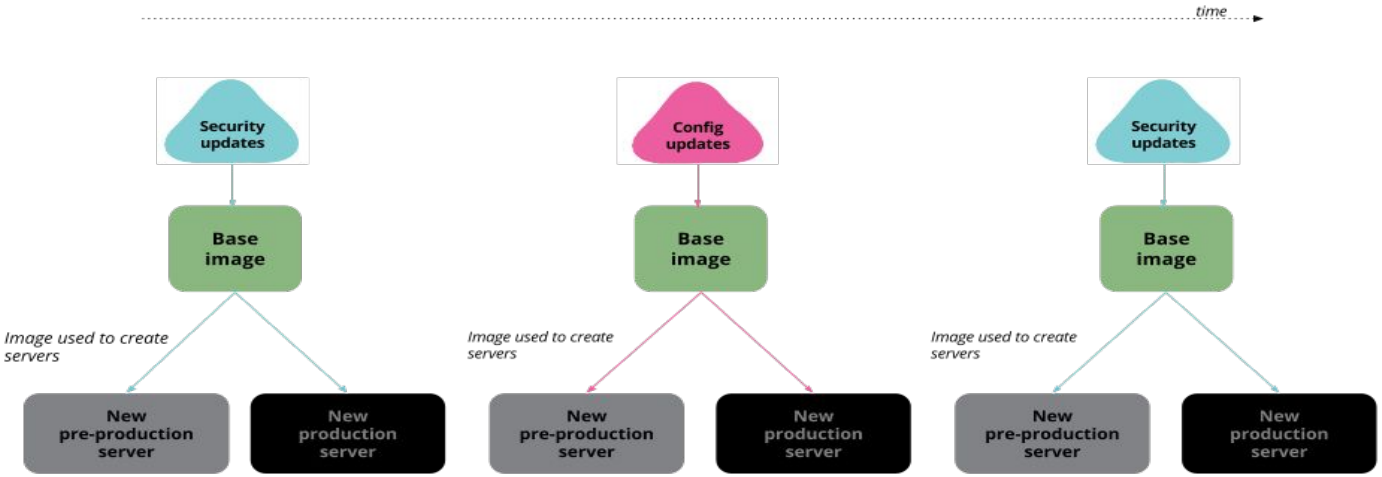
unchanging over time or unable to be changed.
"an immutable fact"
synonyms: fixed, set, rigid, inflexible, permanent, established, carved in stone; unchanging, unchanged, unvarying, unvaried, static, constant, lasting, enduring, steadfast
"the subtext of the liturgy had always been God's immutable power"
antonyms: variable

Phoenix Servers





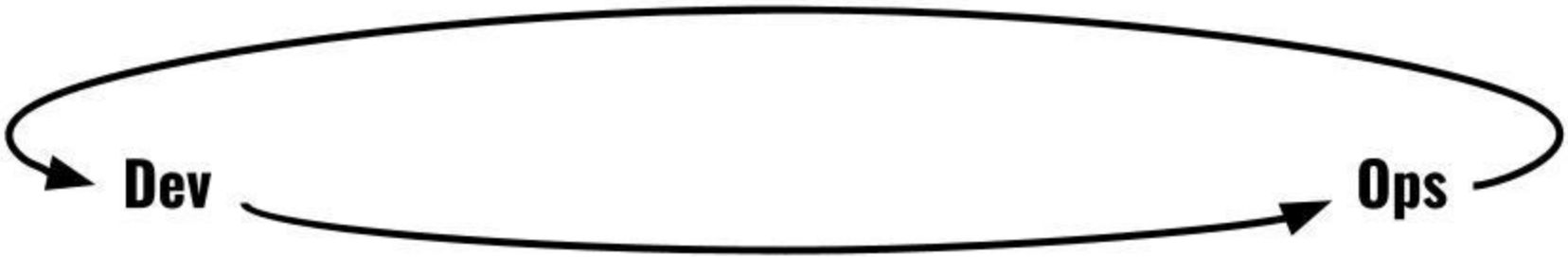
Long running snowflake servers over time

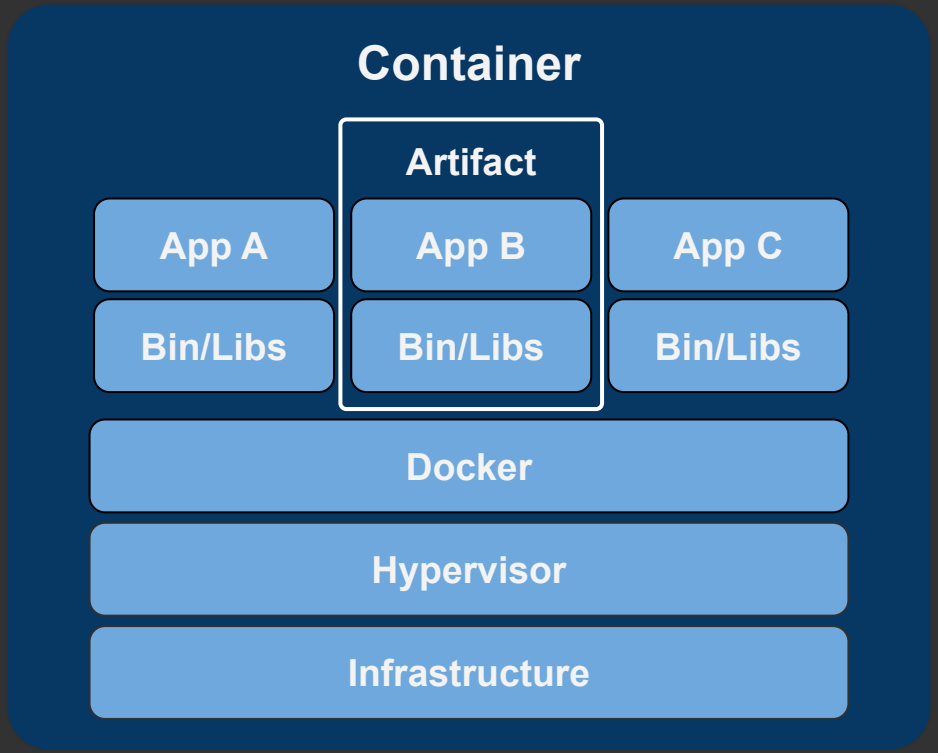


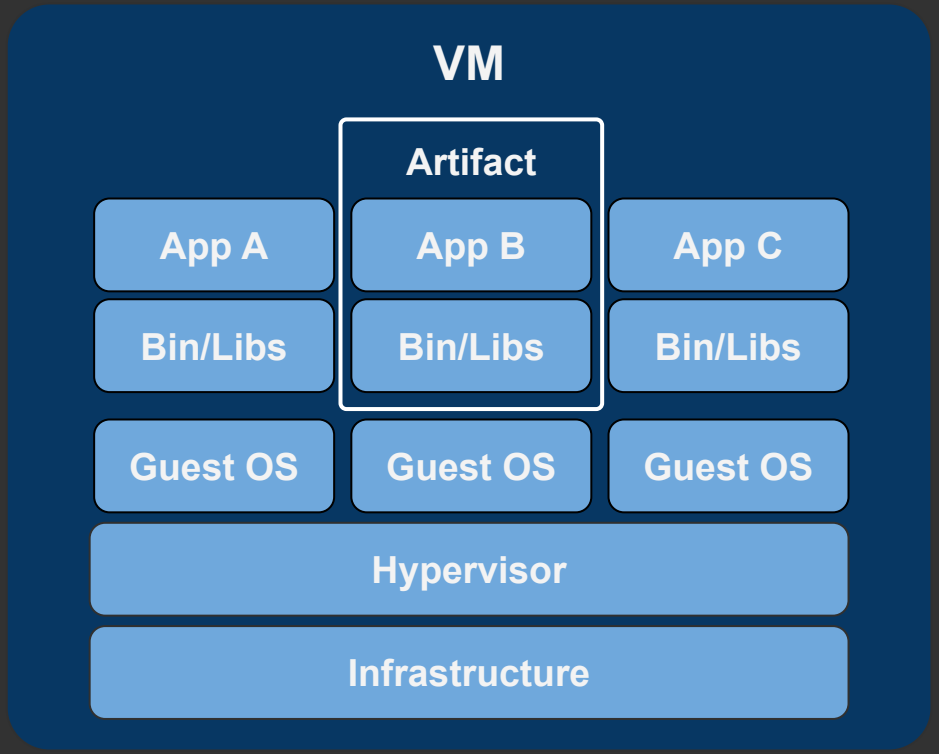
Phoenix servers over time

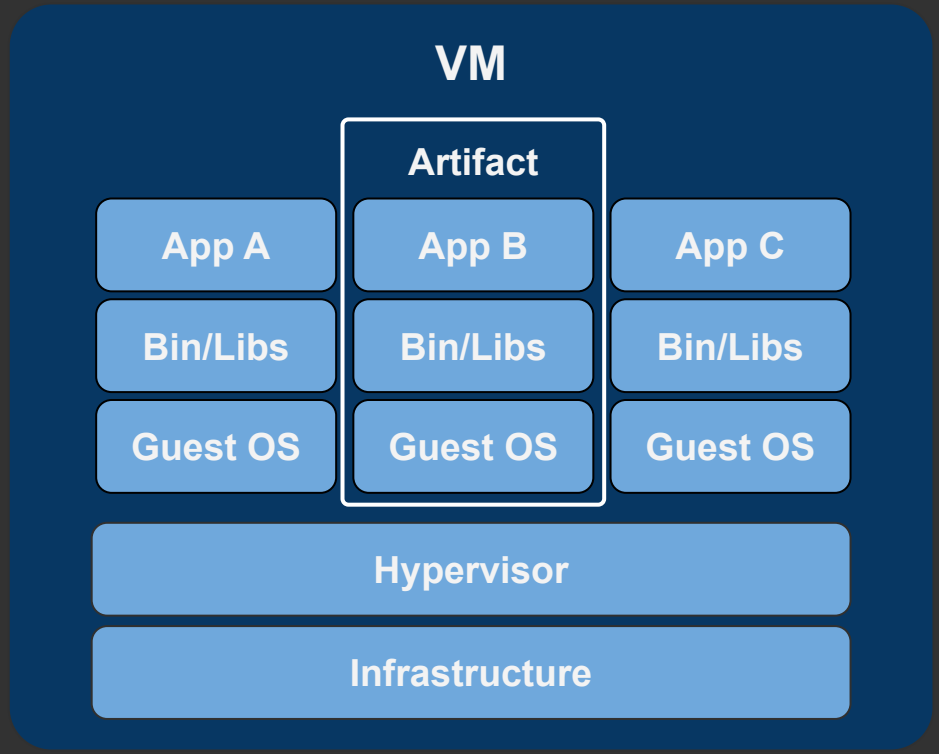
https://www.thoughtworks.com/insights/blog/moving-to-phoenix-server-pattern-introduction

The Second Way:
Amplicação dos Loops de Feedback




Container
Artifact
App A
App B
App C
Bin/Libs
Bin/Libs
Bin/Libs
Docker
Hypervisor
Infrastructure

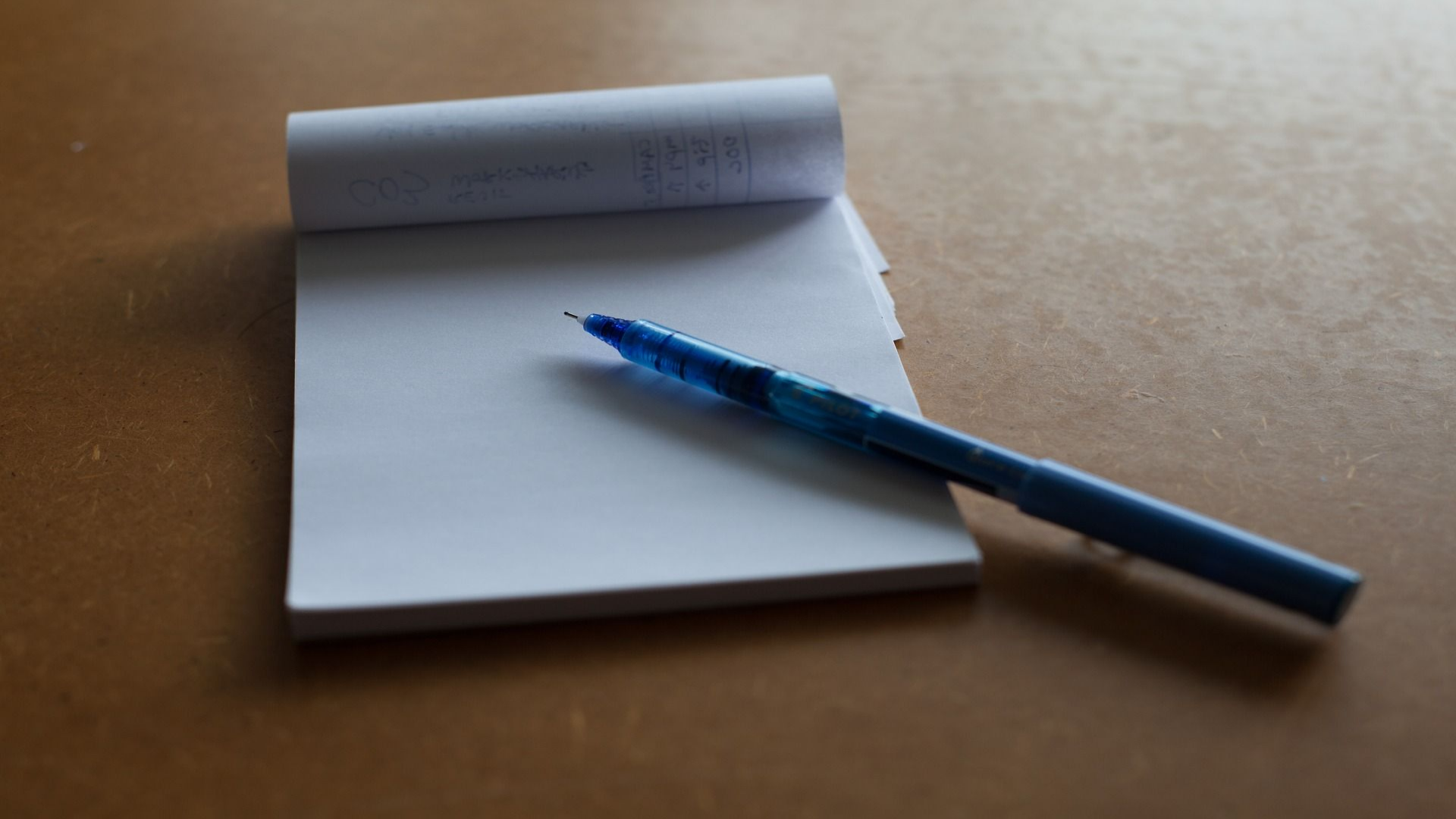

VM
Artifact
App A
App B
App C
Bin/Libs
Bin/Libs
Bin/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


VM
Artifact
App A
App B
App C
Bin/Libs
Bin/Libs
Bin/Libs
Guest OS
Guest OS
Guest OS
Hypervisor
Infrastructure


STOP

Implementação - Não faça em produção

- Atualização de pacotes (bibliotecas e dependências)
- Alteração de configurações
- Modificações na aplicação
- “Não usar SSH”



Implementação - Boas práticas

- Servidores na Nuvem
- Automação completa de todo pipeline de serviços
- Logs centralizados
- Armazenamento de dados em “ambiente” externo
- Equipes de desenvolvimento e operações “engajadas”

Implementação - Boas práticas

- Estado de serviços, dados da aplicação ou qualquer outra informação que deve ser preservada até que outro “servidor” estava “no ar” devem estar em outro local da infraestrutura. (Stateless)

Visibilidade

- Onde e quando foi construído e por que?
- Qual foi a imagem anterior?
- Como iniciar, validar, monitorar e atualizá-la?
- Qual repositório está sendo usado e qual hash do git foi usado para construir a imagem?
- Quais são as tags específicas do container/vm usada como registro do build?
- Qual o nome do projeto no qual pertence o artefato

Como tornar Imutável

- Provisione um novo servidor
- Teste o novo servidor
- Altera a referência para o novo servidor
- Mantenha a versão antiga (temporariamente) para fazer o rollback


NAVY

Pontos de atenção

- Centralização de Logs
- Feature toggle
- Observability
- Múltipla Versões (“temporariamente”) em produção

Pontos de atenção

- Balanceadores de carga com suporte:
  - Canary
  - Blue/Green
  - Rolling Deploys
- Service Mesh
- Boa prática ao usar os códigos de status HTTP

Quando não usar

- Serviços que mantenham o estado localmente
- Serviços Stateful
- Serviços de infraestrutura base

Testes de “aceitação”

- Testes de segurança
- Testes dos serviços
- Testes de conformidade
- Testes de integração

Conclusão

Infraestrutura Imutável é tornar como parte do artefato que será lançado (instalado) em produção a pilha (stack) completa da aplicação. Maximizando os diversos tipos de testes antes do artefato entrar em “produção”


MNF
GIANTMONSTER

Demo

https://github.com/maburix/immutable_infrastructure

Referências

- https://medium.com/netflix-techblog/building-with-legos-d68368fe4ce
- https://martinfowler.com/bliki/PhoenixServer.html
- https://martinfowler.com/bliki/SnowflakeServer.html
- https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure
- https://www.theregister.co.uk/2013/03/18/servers_pets_or_cattle_cern/
- https://boxfuse.com/blog/no-ssh
- https://www.digitalocean.com/community/tutorials/what-is-immutable-infrastructure
- https://www.devopsdays.org/events/2018-sao-paulo/welcome/
- https://blog.buoyant.io/2017/04/25/whats-a-service-mesh-and-why-do-i-need-one/
- https://www.thoughtworks.com/insights/blog/moving-to-phoenix-server-pattern-introduction

Fernando Ike

// linkedin.com/in/fernandoike

// twitter.com/fernandoike