MODERN SOFTWARE DELIVERY

TRIUMPHS AND FAILURES

ZACHARY SCHNEIDER @SIGIL66

- **INSTANA -** OPERATIONS ARCHITECT
  - End game SaaS APM
- **BOUNDARY -** OPERATIONS ARCHITECT
  - Systems and infrastructure monitoring SaaS
- **RACKSPACE -** LEAD OPERATIONS ENGINEER
  - Cloud Databases
  - Cloud Sites
  - Cloud DNS

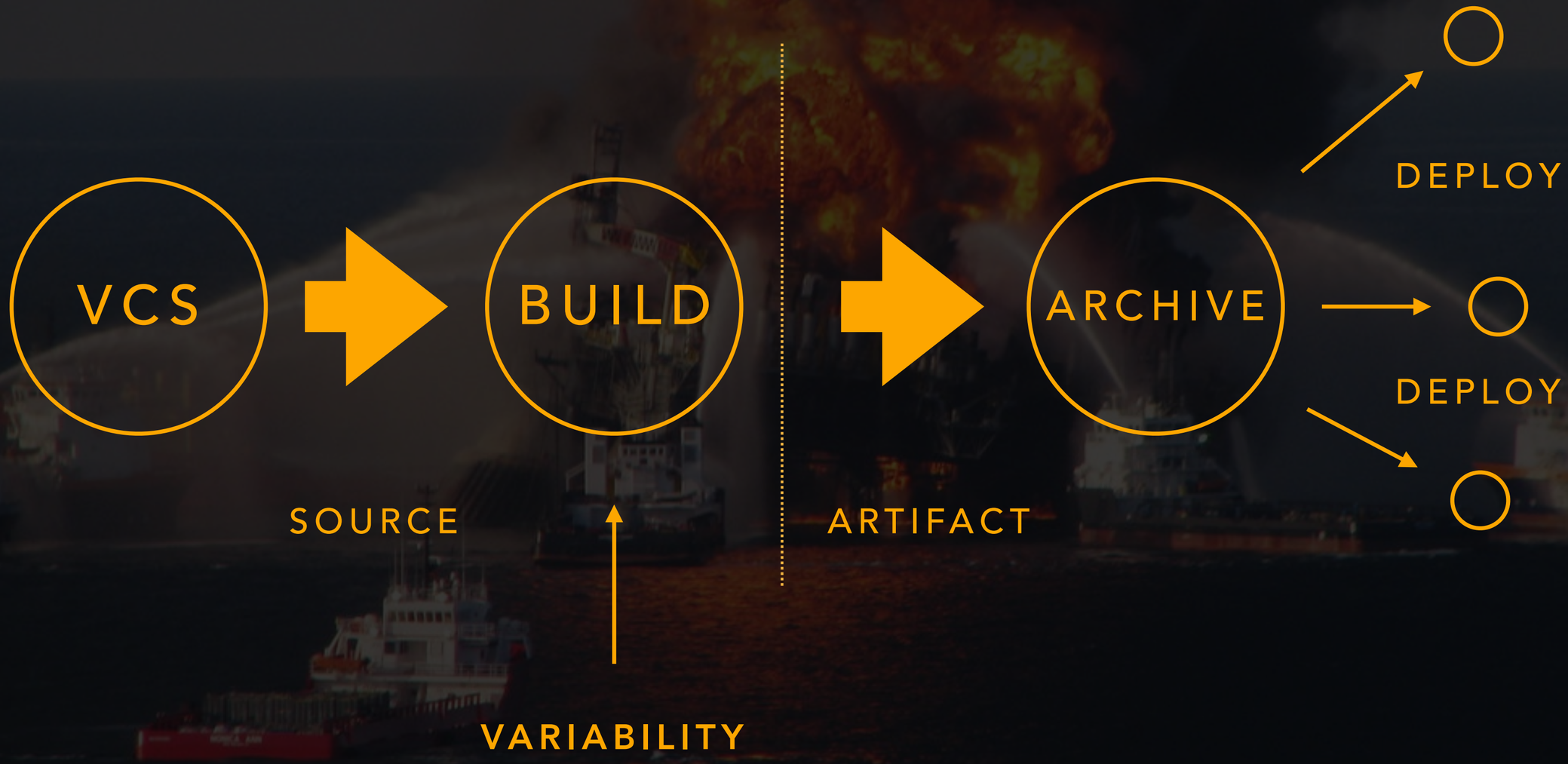
INTRODUCTION

- **MODERN SOFTWARE DELIVERY**
  - A challenging proposition
- **REQUIREMENTS**
  - Create and Enforce Consistency
  - Repeatable Results
  - Auditable Results (security)
  - Must be Usable
  - Must be Operable
    - In house
    - On premise

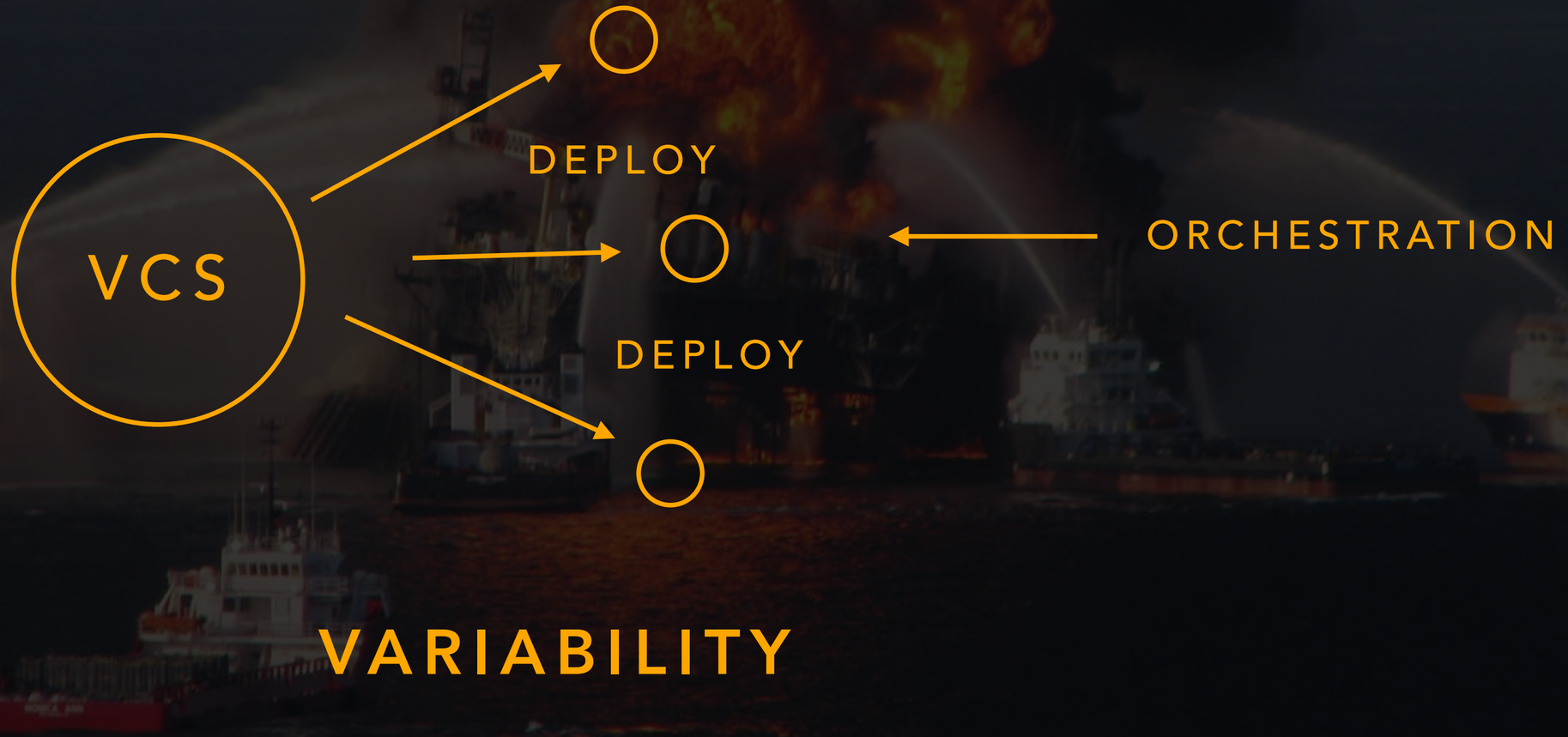
SUMMARY

- COMPONENTS OF SOFTWARE DELIVERY
- ANTIPATTERNS
- PATTERNS
- USABILITY
- GOING BEYOND

COMPONENTS


VCS
BUILD
ARCHIVE
DEPLOY
DEPLOY
DEPLOY
SOURCE
ARTIFACT
VARIABILITY

ANTIPATTERN :: GIT DEPLOY



ANTIPATTERN :: MISC

- **SCPing** binaries around
- **Compiling from source** on non-build systems
- Anything that uses **SSH** as an RPC protocol
  - **YES** this includes ansible

PATTERN :: CONTAINERS (LINUX)

- THE RUNTIME CONTENDERS
  - Docker
  - rkt

PATTERN :: CONTAINERS (LINUX)

- **THE GOOD**
  - Deploying immutable artifacts (consistency)
  - Build infrastructure does not touch deploy environments (security)
  - Tooling is developer friendly … sort of (usability)

PATTERN :: CONTAINERS (LINUX)

- **THE BAD**
  - Dealing with image layers is a nightmare (operability)
    - Space
    - Install time
    - Filesystem drivers
  - Immutability (operability)?
  - Security?
    - selinux
    - apparmor
    - seccomp
    - non root == exercise in futility due to UID mapping
  - Container builds typically a mess (usability)

PATTERN :: PACKAGES

- THE OLD NEW STUFF
  - **pkgsrc** - 1997
  - **apt** - 1998
  - **yum** - 2003
  - **IPS** - 2008

PATTERN :: PACKAGES

- **THE GOOD**
  - Deploying immutable artifacts (consistency)
  - Build infrastructure does not touch deploy environments (security)
  - Systems composed of packages can be cryptographically verified and repaired (security)
  - Overall easy to operate and maintain (operability)
  - Small frequent updates are very fast (operability)

PATTERN :: PACKAGES

- **THE BAD**
  - Package build methodology is coupled with the package system
  - Package and Repository metadata formats are not easily serializable/ accessible
  - Package file formats: WTF?
  - Requires tech layering to deal with usability issues
    - FPM
  - Package versioning for various systems **NOT** driven by a single standard

USABILITY :: SYSTEMS SOFTWARE

- Layering usability on top of archaic solutions
- Examples
  - **IPTABLES**
    - ufw
    - firewalld

USABILITY :: SYSTEMS SOFTWARE

- **Good vs Bad**
- **Examples**
  - ZFS vs Btrfs
  - OpenBSD PF vs IPTables
  - Solaris Crossbow vs OpenVSwitch
  - OpenBSD ipsecctl vs FreeSWAN

USABILITY :: SYSTEMS SOFTWARE

IPTABLES

iptables -A INPUT -p tcp -s 10.0.0.1/32 —dport 22 -j ACCEPT

vs

OpenBSD PF

pass in proto tcp from 10.0.0.1/32 to port ssh

GOING BEYOND :: REQUIREMENTS

- Create and Enforce Consistency
- Repeatable Results
- Auditable Results (security)
- Must be Usable
- Must be Operable

GOING BEYOND :: ZPS

- **ZPS** - The last word in package management
- **Influencers**
  - APT
  - IPS
  - NPM

ZPS :: PRINCIPLES

- Satisfies requirements of modern software delivery
- Developer and operator ease of use
- Support multiple operating systems
- System state can be modeled as a set of packages
- Software installation/deinstallation can be modeled as a set of actions
- No custom metadata formats in packages or repositories
- Package versions utilize a defined standard combined with a timestamp component

ZPS :: PRINCIPLES

- Repositories are by default multi vendor
- Packages in repositories can be added to channels
- Systems can be configured to update from a repository pool or channel
- It should be easy and **fast** to validate and repair the integrity of an installed package set
- A developer should never have to adopt a build system to build a package
- Package file format should be accessible to developers
- Images for container systems should be easy to generate from a list of packages

ZPS :: PRINCIPLES

- Multiple installation roots can be supported by environment manipulation not filesystem manipulation
- Installation policies may be defined and enable with a policy switch
- Should complement Hashicorp tool set for future integration

ZPS :: MILE STONE 1

- All features required to operate as an ancillary package manager
- Multiple ZPS roots
- Linux, OSX, SmartOS
- Actions implemented:
  - Meta
  - File, Dir, Symlink
  - Signature
  - Depend
  - User
  - Group
  - Service

ZPS :: MILE STONE 2

- All features required to operate as a root package manager
  - Illumos
  - Linux
  - FreeBSD
- Actions
  - File - add tags for custom behavior
  - Template

QUESTIONS?