SALT CONF 17

A Demonstration of SaltStack Orchestration for Docker Containers

I needed a really long title for something so small

SALTSTACK

SALT CONF 17

A Demonstration of SaltStack Orchestration for Docker Containers to act like a serverless infrastructure or Function as a Service (FaaS)

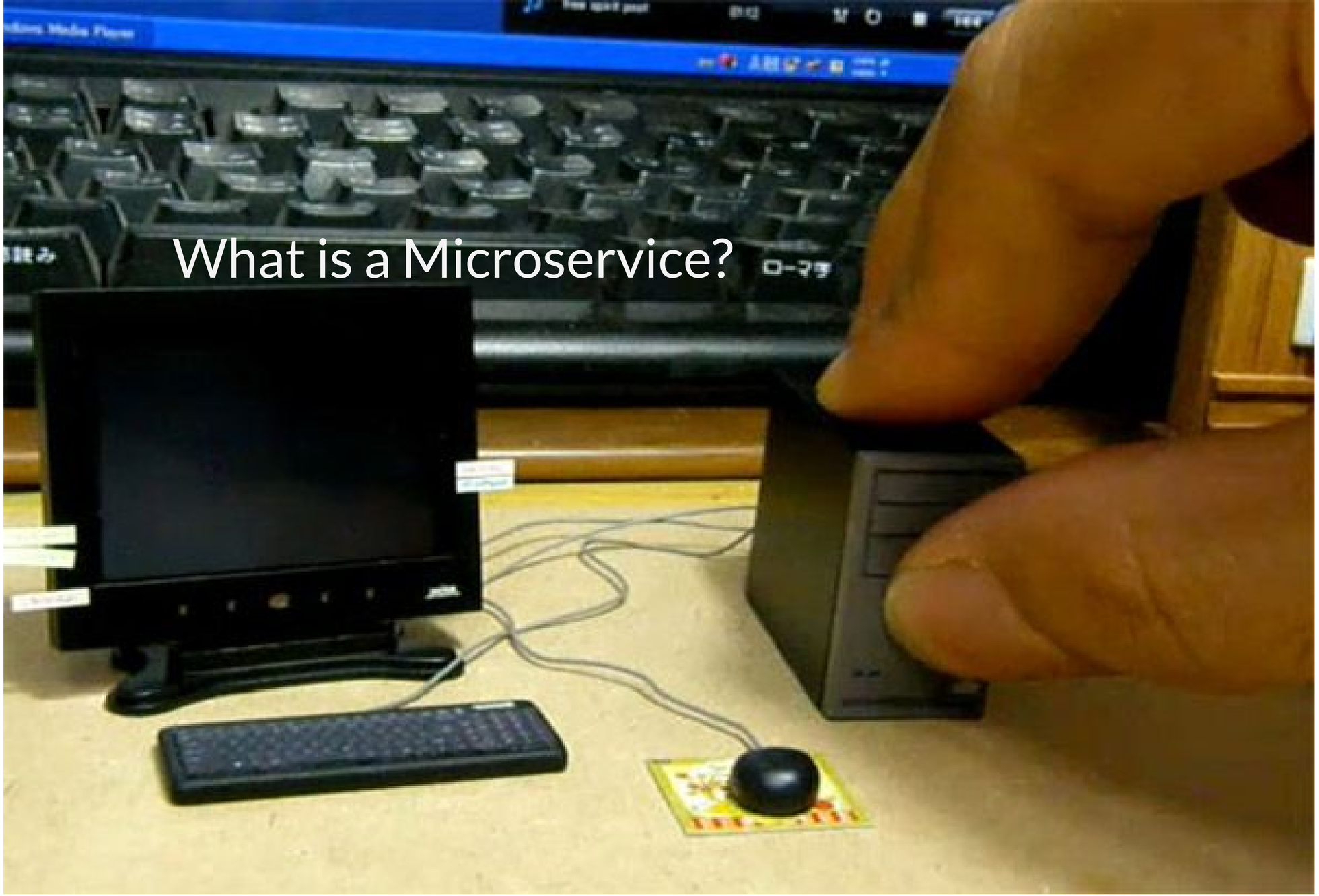
SALTSTACK

Charles C. Ruffino

SaltStack Solutions Consultant - AKA Pre-Sales Engineer

ITS
COMPAQ
DELL
AppSense
altiris
Aegis MORTGAGECORPORATION
Bay Dynamics
Productivity POINT INTERNATIONAL
Symantec
TEKsystems
SALTSTACK

What is a Microservice?



Greg Coleman Follow
Devops . Sys Admin . Foodie . Polyglot
Nov 14, 2016 · 3 min read

Writing Docker Microservices in COBOL

MICROSERVICES

NaCl

A microservice is a tightly scoped, strongly encapsulated, loosely coupled, independently deployable and independently scalable application component. Based on a combination of SOA and domain-driven design (DDD), MSA is a design paradigm that has three core objectives: development agility, deployment flexibility and precise scalability.

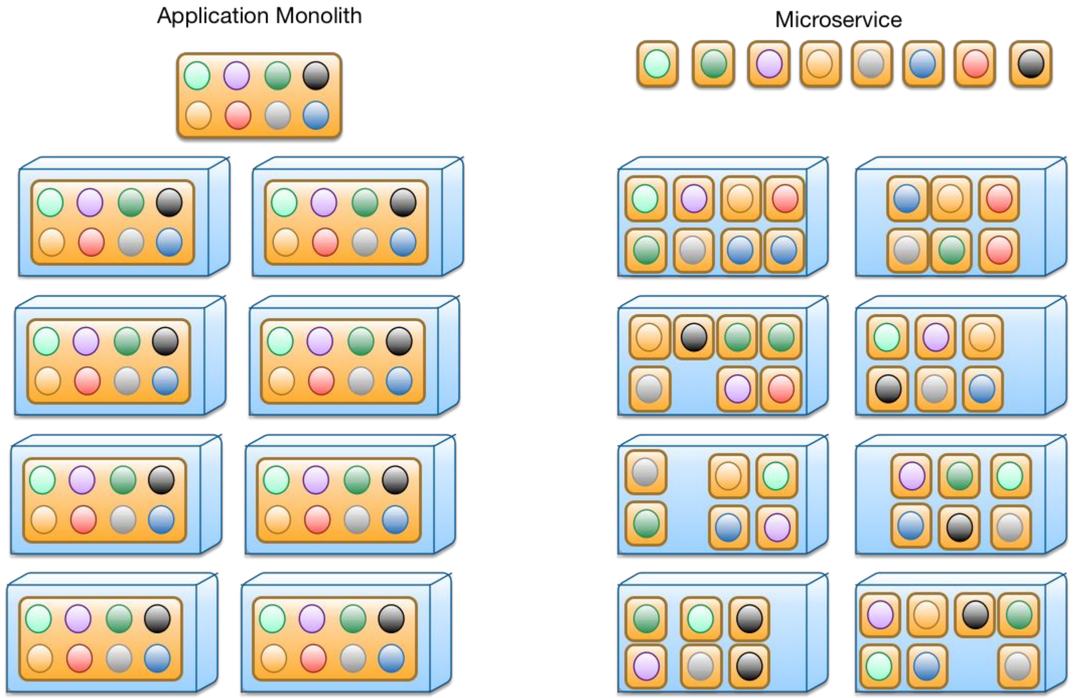
Gartner, Inc

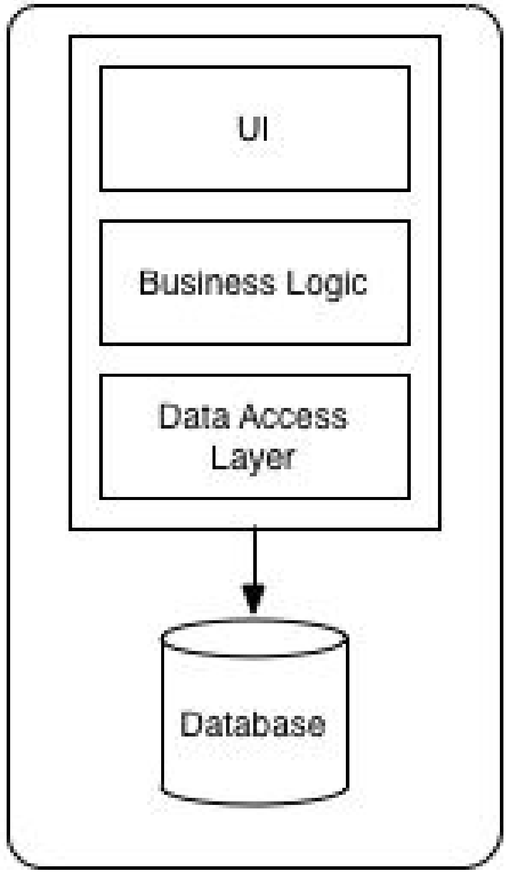
MICROSERVICES

$MgSO_4$

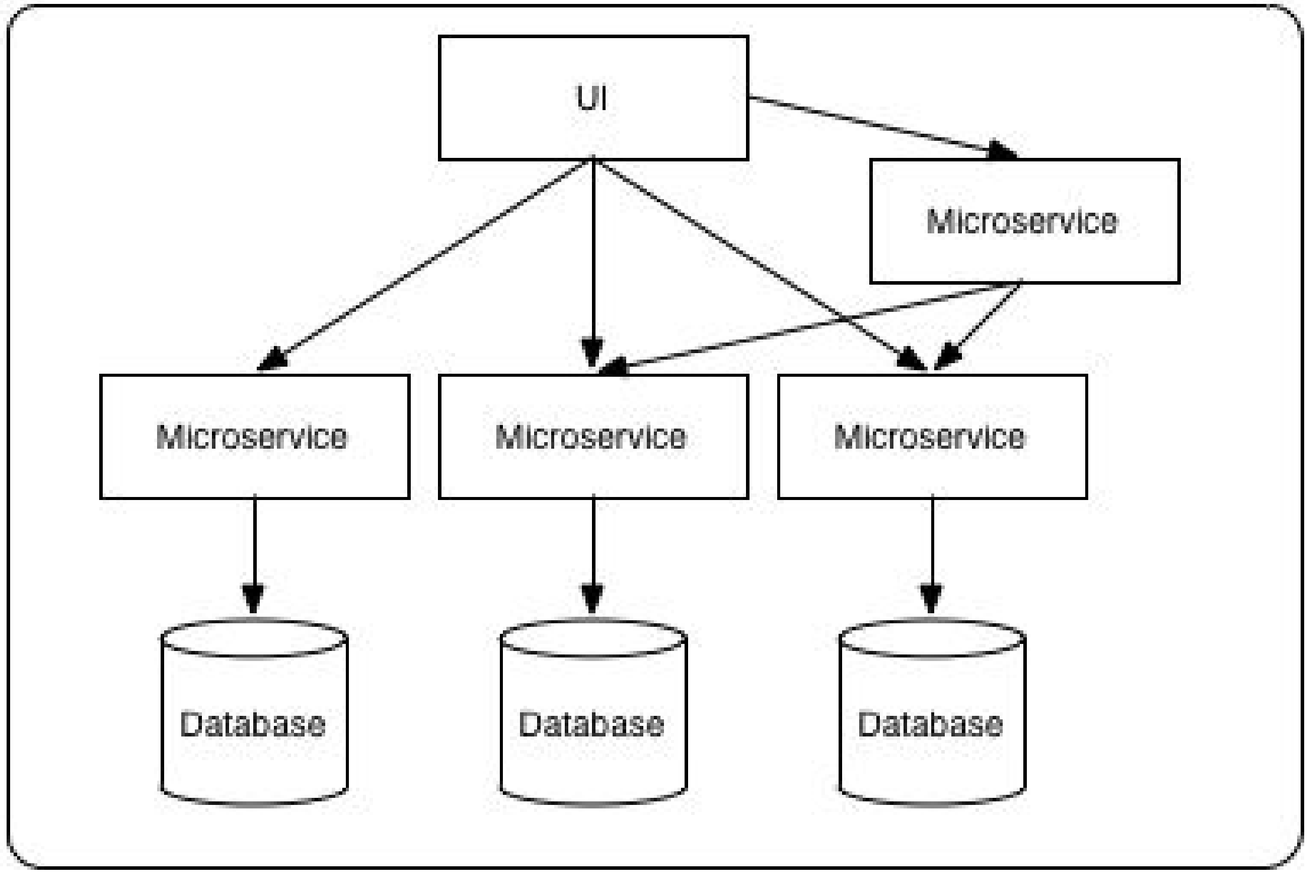
Martin Fowler

In short, the microservice architectural style is an approach to developing a single application as a suite of small services, each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment machinery. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.





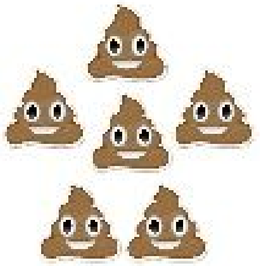
Monolithic Architecture



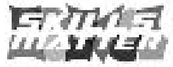
Microservices Architecture

Monolithic vs Microservices

Monolithic



Microservices



@alvaro_sanchez

odobo

MICROSERVICES

KI

Shimshon
Zimmerman

Microservices architecture is a loosely coupled Service Oriented Architecture (SOA) with bounded contexts. In practice, this means a microservice only **does as few things as necessary** and **works within as limited an area of the domain as possible**. In other words, it helps **isolate functions and uses asynchronous communications**, making it independent and as simple as possible.

MICROSERVICES or SERVERLESS

$CuSO_4$

Shimshon
Zimmerman

Microservices architecture is a loosely coupled Service Oriented Architecture (SOA) with bounded contexts. In practice, this means a microservice only **does as few things as necessary** and **works within as limited an area of the domain as possible**. In other words, it helps **isolate functions and uses asynchronous communications**, making it independent and as simple as possible.

Serverless architecture leverages third-party services such as Lambda to eliminate the need to establish and maintain the traditional infrastructure sitting behind an application. This approach can significantly reduce time to market as well as operational cost and complexity.

Elements of Microservices

$CaCl_2$

or really, “Characteristics”

Each functional element is a separate service

All elements are deployed independently

All elements are deployed across servers and replicated as needed

Elements of Serverless

$KMnO_4$

... but it works on my box

Operational responsibility is abstracted away to a 3rd party

Code runs with a series of entry hooks and exit hooks

Other services are exposed as integration end-points

Services are connected to each other not through directed API endpoints but through events.

This model may grow from its existing ecosystem

Allows for a very flexible approach for developers

Arthur C. Clarke

CH_3COONa

"That thing was real!?!"

When a distinguished but elderly scientist states that something is possible, he is almost certainly right. When he states that something is impossible, he is very probably wrong.

The only way of discovering the limits of the possible is to venture a little way past them into the impossible.

Any sufficiently advanced technology is indistinguishable from magic.

Elements of Salt

$(CH_3COO)_2Ca$

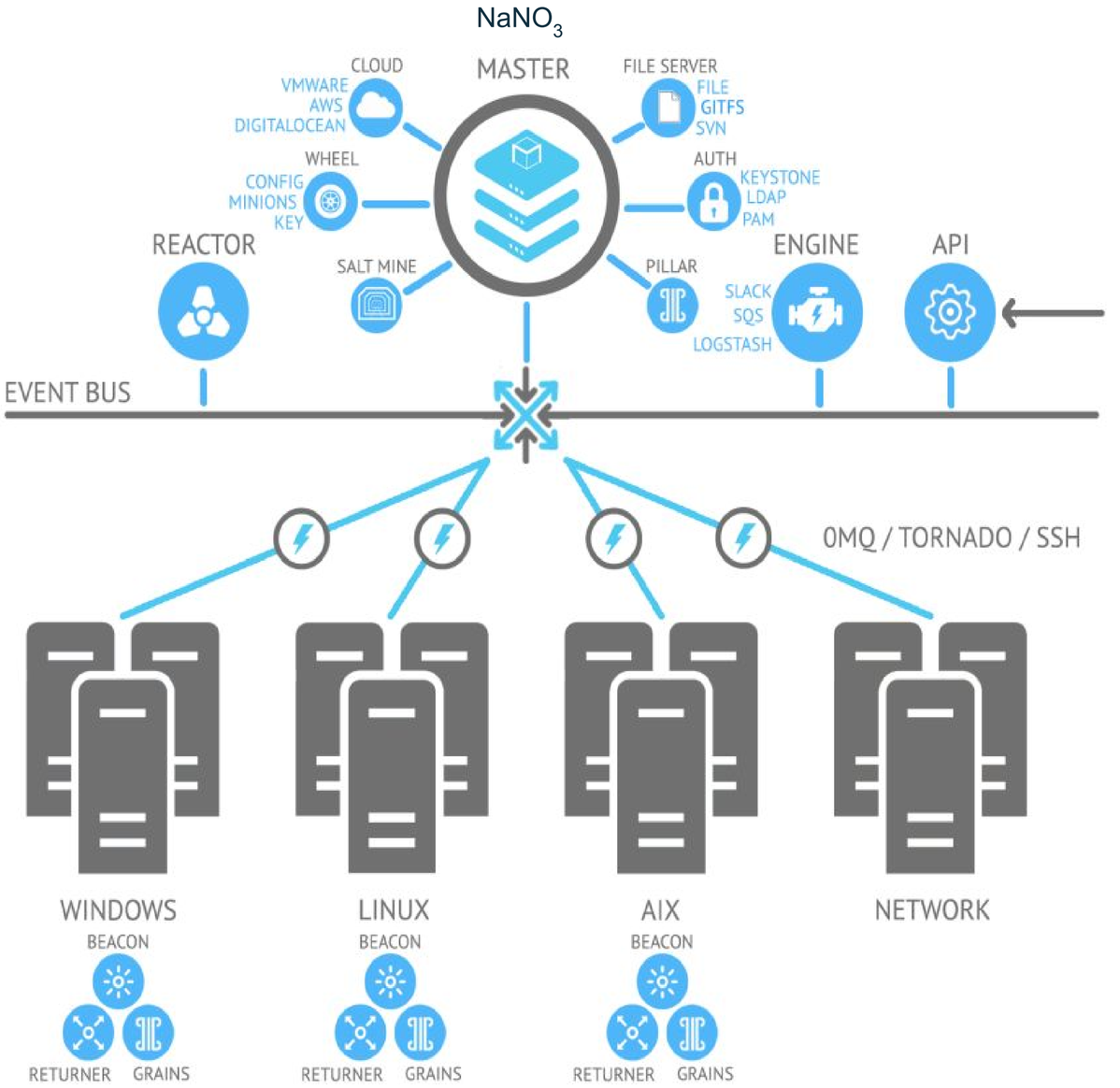
SaltStack is a high-speed platform for provisioning and controlling distributed computing resources

A secure high-speed communications bus that can send targeted publications to connected hosts and have them act on those publications.

A configuration management engine to provide idempotent application of machine states to those hosts.

A reactive backplane for all types of system and application events and a programmable interface to allow for reactions to specified events to trigger actions.

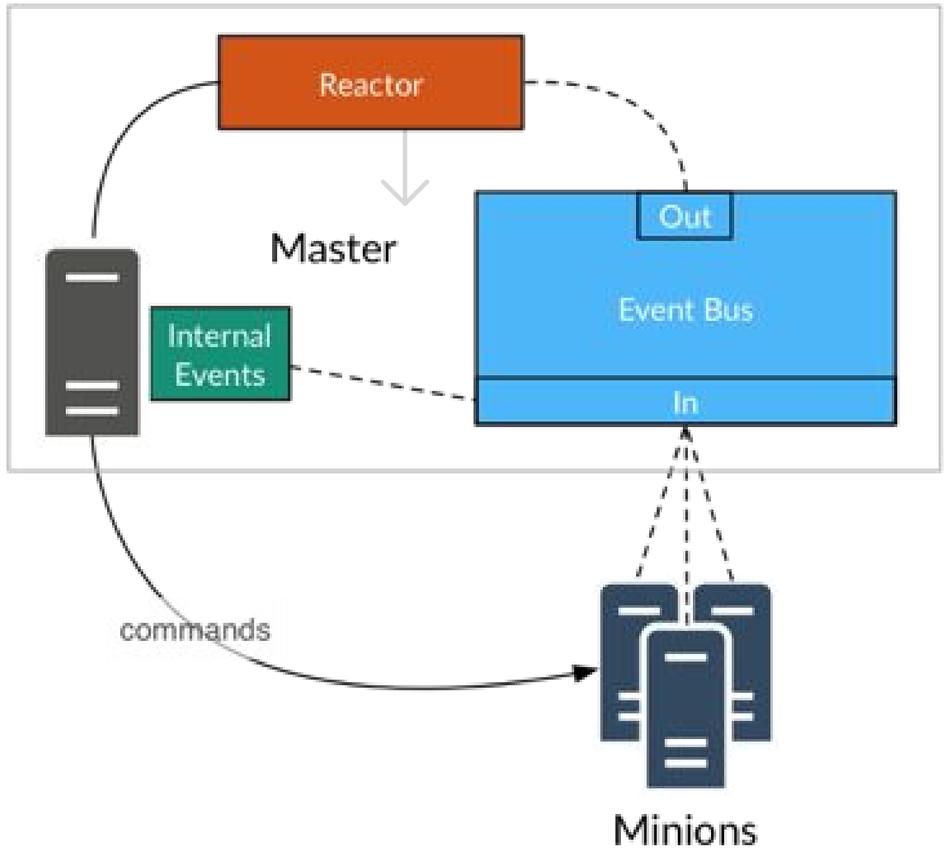
Elements of Salt



Elements of Salt

KNO_3

Reactor: Trigger actions in response to an event.



Elements of Salt

$Al(NO_3)_3$

Engines: looooooooooooong-running, external system processes that leverage Salt

`docker_events`	Send events from Docker events
`hipchat`	An engine that reads messages from Hipchat and sends them to the Salt event bus.
`http_logstash`	HTTP Logstash engine
`ircbot`	IRC Bot engine
`junos_syslog`	Junos Syslog Engine
`logentries`	An engine that sends events to the Logentries logging service.
`logstash`	An engine that reads messages from the salt event bus and pushes them onto a logstash endpoint.
`napalm_syslog`	NAPALM syslog engine
`reactor`	Setup Reactor
`redis_sentinel`	An engine that reads messages from the redis sentinel pubsub and sends reactor events based on the channels they are subscribed to.
`slack`	An engine that reads messages from Slack and sends them to the Salt event bus.
`sqs_events`	An engine that continuously reads messages from SQS and fires them as events.
`stalekey`	An engine that uses presence detection to keep track of which minions have been recently connected and remove their keys if they have not been connected for a certain period of time.
`test`	A simple test engine, not intended for real use but as an example
`thorium`	Manage the Thorium complex event reaction system
`webhook`	Send events from webhook api

Demo Time

magic time

Conclusions

$Al_2(SO_4)_3$

Serverless is a pattern

What is happening in FaaS isn't a provider-based service but instead is an infrastructure design pattern.

Move up the stack

Using an automation framework CFM just to provision and configure machines is missing much of the picture.
Automation can be a part of the application runtime!

Runtime Environments

Automation is more than just configuring system state, it is about configuring runtime environments for systems. Containerization lets us target applications to runtimes instead of having to ship the services _in_ the containers, as we do with microservices.

Questions?

Thanks for coming!

https://articles.microservices.com/writing-docker-microservices-in-cobol-de72d5947b4f
https://martinfowler.com/articles/microservices.html
https://www.stratoscale.com/blog/cloud/aws-lambda-features-limitations-practical-examples/
https://www.stratoscale.com/blog/compute/keeping-small-serverless-functions-vs-microservices
https://rubygarage.org/blog/advantages-of-using-docker-for-microservices/
https://medium.com/aws-activate-startup-blog/using-containers-to-build-a-microservices-architecture-6e1b8bacb7d1
https://www.stratoscale.com/blog/compute/keeping-small-serverless-functions-vs-microservices/
https://www.stratoscale.com/blog/cloud/aws-lambda-features-limitations-practical-examples/
https://blog.algorithmia.com/algorithms-as-microservices/
https://www.ibm.com/blogs/bluemix/2017/06/5-principles-of-microservices-operations-management/
https://howtodoinjava.com/microservices/microservices-definition-principles-benefits/

Name	Chemical formula
Sodium chloride	$NaCl$
Magnesium sulfate (Epsom salt)	$MgSO_4$
Potassium iodide	KI
Copper sulfate	$CuSO_4$
Calcium chloride	$CaCl_2$
Potassium permanganate	$KMnO_4$
Sodium acetate	CH_3COONa
Calcium acetate	$(CH_3COO)_2Ca$
Sodium nitrate	$NaNO_3$
Potassium nitrate	KNO_3
Aluminium nitrate or Aluminum nitrate (US)	$Al(NO_3)_3$
Aluminium sulfate or Aluminum sulfate (US)	$Al_2(SO_4)_3$