Tali Soroker

Building the Perfect Enterprise DevOps Stack

A Complete Guide

Table of Contents

Introduction

Things Are Changing... Quickly

Over the past few years, we have experienced a shift in the way we deliver applications. We are being forced to **innovate faster and deliver higher quality software**. In response to this, the way we approach the design and delivery of software has changed.

Most would identify the cloud as a main component of this change, but it is too abstract of a concept to capture the true depth at which we've changed. As you unravel the layers of change, what you will find is our entire thinking of how we get our solutions in front of customers has also transformed.

The most impactful change is the application of supply chain concepts to software. We have created a **software delivery supply chain - a method to plan, implement, and control the efficient, effective storage and flow of software from concept to customer**.

The supply chain concept identifies process improvement to efficiently transform raw materials into physical product and has evolved over time to include concepts of integration, globalization and specialization. This original evolution (for hard products) took decades. **In software, the supply chain evolution has taken just a few years.**

Chapter 1

The Modern Software Delivery Supply Chain

The Bottom Line? Everything has changed

Everything we build today and the tooling we employ is typically built to an API. This API movement originated as a vehicle to create platforms where others could build on top of a framework. Today, this API movement is more of a complex web of interactions where all the building blocks work together through an agreed upon standard for a common goal. We have building blocks that all work together and this force more than any has contributed to the emergence of the new supply chain that is comprised of a few groups of capabilities, which includes: Workflow, Deployment and Monitoring.

However, before we get into the three groups, we should note that just as there is raw material that enter a physical supply chain (steel, wood, plastic, etc…) there is also a set of raw material that enters our software supply chain. This too has changed over the past few years as open source software (OSS) has become the commodity and the baseline of almost every application. It would be rare to find a code base that didn't include OSS in some way. And with OSS much of our new raw materials have become abundant and they evolve faster than ever. Now, let's explore the tools.

Product Creation: Dev Collaboration and Workflow Tools

The same tools that fueled the emergence of OSS are also transforming our dev teams. Pull requests and Jira tickets are the lingua franca for the modern development organizations. Agile development, collaboration and instant information have changed the game. It's the way software is built. We also have a new way of packaging our apps so we can enjoy maximum portability, containers.

Some of the new tooling in this part of the supply chain includes:

- Source control and collaboration: Github, Bitbucket, Gitlab
- Issue tracking and agile: Jira, ServiceNow
- Packaging: VMWare, Docker
- ChatOps: HipChat, Slack

Product Delivery: Deploying our Applications

A major shift in the arm for speed of delivery of applications has been driven by a need to automate common processes. Building software and deploying infrastructure is typically an exacting and detailed task that many operators had built scripts around. Over the past few years, tooling has emerged to take these common processes and wrap them with tooling. We have even started to deliver new platforms to automate orchestration of workloads.

Some of the key tools here include:

- CI/CD: Jenkins, Travis CI, Circle CI, TeamCity, Codeship, GitLab CI, Bamboo
- Provisioning, Configuration Management: Chef, Puppet, Ansible, Terraform
- Orchestration: Kubernetes, Mesos, Pivotal Cloud Foundry

Product Success: Monitoring and Reporting

Once applications and services are live we must make sure they not only do what they do, but they perform and deliver as advertised. A set of tools has been created and adopted to take care of this. We have log aggregators and log analysis tools that allow us to search through this information to determine what our apps want to tell us. We can also monitor our apps for slowdowns and performance degradation so we can ensure we meet customer demands. And open source stacks have evolved so we can send data to tools and run analytics and dashboards as well.

Some of this tooling includes:

- Log Management: Splunk, SumoLogic, ELK, Graylog, Loggly
- APM: AppDynamics, New Relic, Dynatrace, and open source flavors like Kamon
- Alerting & Visualization: PagerDuty, Pingdom, Graphite, Grafana

Then and Now: Monolith and Microservices

The main difference between the 20th century “hard” product supply chain and that which we use for software today is we are shifting away from conceptualizing and maintaining a single product, end to end to creating individual components that communicate through an agreed upon language, these APIs. And it’s not just hard goods, it’s how we are breaking own our monolithic application into microservices.

This new approach employs individual and unique blocks of code that are all sourced and maintained within their own paths but ultimately integrate into a whole. It’s as if each lego block in a package is sourced, built and delivered all separately. The line tooling is the same but the raw materials differ and in the end the way we assemble into a final product.

The software supply chain has ushered in this new approach. It is a direct influence in the move to microservices.

Supply Chain Optimization Doesn't Come Without Challenge

Ultimately, our development teams are being pushed to the limits as we use this new efficient supply chain to push more software into production at an accelerated rate. But this doesn't come without a cost. As we speed adoption of delivery across smaller components we increase complexity and introduce significant risk. There is an opposing force to speed; and that is reliability. Quality control in the software supply chain is complex.

At OverOps, we exist to address this problem. We believe the delivery of reliable software should be a primitive for every organization. Reliable software should work the way it's supposed to work, perform as expected and be secure. We help you make sure it does what is it is supposed to do. We help you with functional quality.

Chapter 2

Developer Collaboration and Workflow Tools

product creation

A successful application begins in the early stage with effective collaboration and communication between team members

Successful software starts with a strong workflow and solid communication. For team members across the delivery supply chain, tools that help easily fork new code and track code sources is important to ensure efficiency and code quality. Going along with that is communication, which may be one of the most crucial elements of a strong engineering team. Without proper communication channels, code can more easily slip by unreviewed, tasks and team updates can get lost in the shuffle.

Workflow: GitHub, GitLab, BitBucket

GitHub

GitHub is pretty comfortably the frontrunner in this space in terms of user volume (and thus community support) and code volume. They strongly support the open-source community and open-source projects by, not only allowing unlimited free public repos, but charging for private repos.

BitBucket

BitBucket is a strong competitor to GitHub, but has, understandably, chosen to advance in a different direction that its predecessor. BitBucket is more focused on catering to enterprise developers, especially after its acquisition by Atlassian in 2010. BitBucket offers a free version for teams of up to 5 developers including the private repos and, another benefit for relevant users, integration to other Atlassian products like Jira.

GitLab

GitLab is another git-based repository hosting platform with some key differences separating it from GitHub and BitBucket. GitLab basically took the basic idea of repo-hosting and infused it with more fine-tuned features to support team collaboration. For example, they boast different authentication levels for different team roles and built-in CI/CD service.

Communication (+ ChatOps): HipChat (Stride), Slack

HipChat (Stride)

HipChat is a popular platform from Atlassian that was designed specifically for business team communications. Along with direct and group chat capabilities,

HipChat also integrates video chat so team members working in different locations can meet face-to-face in the same platform.

Stride is a newly released product from Atlassian that they recommend teams upgrade to and that will likely replace HipChat entirely in the future (though no plans for deprecation of HipChat have been announced). Stride was designed to be a more formidable competitor to Slack with new and enhanced messaging and communications offerings.

It offers many of the same messaging features as Slack like direct messages and group chats, but also adds video-chat and screen sharing into the mix. For teams working with other Atlassian products, it's hard to deny the benefit of being able to access everything without toggling between multiple systems.

Slack

Slack is quickly becoming one of the most popular messaging services for modern office communication. It's an excellent system for direct messaging, team collaboration and company-wide communications. Plus, with all of the apps and bots that you can create and use, it has endless potential.

For teams using Slack, it's already a significant and valuable part of the software development process by improving and simplifying internal communication. Integrating with external monitoring tools extends its influence to the error tracking and resolution part of the SDLC by aggregating alerts and facilitating communication for faster resolution times.

Chapter 3

Continuous Integration, Delivery and Deployment Tools

Rapid codebase growth and the acceleration of software delivery contributed to the rise of these Continuous Integration and Continuous Deployment/Delivery tools

The use of CI/CD tools automates the process of building, testing and deploying new code. Each team member can get immediate feedback about the production readiness of their code, even if they just changed a single line or character. That way, each team member can push their code into production, while the process of building, testing and deploying is done automatically so they can move on to work on the next part of the application.

Jenkins

Jenkins is one of the more known and common names in the CI market. It started as a side project by one of Sun's engineers, and expanded into one of the biggest open source (i.e. *free)* CI tools that helps engineering teams automate their deployments. Full disclosure: we at OverOps also use Jenkins, along with a homegrown CLI tool.

Just like a CI tool promises, with Jenkins you can automate your build, test and deploy tasks. The tool supports Windows, Mac OSX and various Unix systems, and can be installed using native system packages, as well as Docker or installed as a standalone on any machine with a Java Runtime Environment (JRE) installed.

On the practical side, Jenkins gives any member of the team the ability to push their code to the build, and get immediate feedback on whether it's ready for production or not. In most cases, this will require some tinkering and tailoring of Jenkins according to your team's custom requirements.

The place where Jenkins shines is with its rich plugin ecosystem. It offers an extended version with over 1,000 plugins, which allows integration with almost every tool and service that's available on the market. Being an open source tool also gives you the option to custom fit it for a home-grown solution, just like we did. However, the need to spend time and some effort to make sure it is suitable for you might be a downside for some teams.

Plus, open source + plugins = community. Any configuration, workflow, need or desire you can think of, you'll have an option to create it with the help of Jenkins and its plugins. Also, great name for a band.

If you're looking for a cheap (free!) CI solution, are willing to put in the work to customize your environment and need support from a community of users, Jenkins is the choice for you.

Travis CI

Travis CI is one of the more common names in the CI/CD ecosystem, created for open source projects and then expanded to closed source projects over the years. It's focused on the CI level, improving the performance of the build process with automated testing and an alert system.

Travis-CI focuses on allowing users to quickly test their code as it's deployed. It supports large and small code changes, and is designed to identify changes in building and testing.

When a change is detected, Travis CI can provide feedback whether the change was successful or not.

Developers can use Travis CI to watch the tests as they run, run a number of tests in parallel, and integrate the tool with Slack, HipChat, Email and so on to get notified of issues or unsuccessful builds.

Travis CI supports container builds, Linux Ubuntu and OSX. You can use it across different programming languages, such as Java, C#, Clojure, GO, Haskell, Swift, Perl and much more. It has a limited list of third-party integrations, but since the focus is on CI rather than CD, this might not be an issue for your use case.

Basically, if your code is open source and you're more concerned about the continuous integration of your builds, Travis CI is worth a try.

Circle CI

Circle CI is a cloud based tool that automates the integration and deployment process. It also focuses on testing every change to the code before it's deployed, using a number of methods such as unit tests, integrations tests and functional tests. The tool supports containers, OSX, Linux and can run within a private cloud or your own data center.

Circle CI integrates with your current version control system, such as GitHub, Bitbucket and others, and runs a number of steps whenever a change is detected. These changes could be commits, opening PRs or any other change to the code.

Every code change creates a build and runs the tests in a clean container or in a VM, according to your initial configurations and preferences. Each build consists of a number of steps, including dependencies, testing and deployment. If the build passes the tests, it can be deployed through AWS CodeDeploy, Google Container Engine, Heroku, SSH or any other method of your choice.

The success or failure status of the builds and tests in question is sent via Slack, HipChat, IRC or a number of other integrations, so the team can stay updated. It's important to note that Circle CI requires some tweaks and changes for a number of languages, so it's best to go over the documentation for your language of choice.

Plus, Circle CI can auto-cancel redundant builds on GitHub. If a newer build is triggered on the same branch, the tool identifies it and cancels the older builds that are running or queued, even if the build in not finished.

Basically, if you're looking for a GitHub friendly tool with a wide community behind it, that can also run within a private cloud or your inside own data center, Circle CI is worth checking out.

TeamCity

TeamCity is a CI/CD server, made by JetBrains. It offers continuous integration “out of the box”, as well as allowing users to best fit the tool to their own needs. It offers support for a number of languages (Java, .NET, Ruby and others), and has JetBrains to back the tool up support and documentation wise.

As a CI/CD tool, TeamCity aims to improve release cycles. With it you can review on-the-fly of test results, see code coverage and find duplicates, as well as customize statistics on build duration, success rate, code quality and other custom metrics.

Once TeamCity detects a change in your version control system, it adds a build to the queue. The server finds an idle compatible build agent and assigns the queued build to this agent, which executes the build steps.

While this process is running TeamCity server logs the different log messages, test reports, and other changes that are being made. The changes are saved and uploaded in real time, so users can know what’s happening with the build as they make changes to it. The tool also offers an option to run parallel builds simultaneously on different platforms and environments.

It also comes with an option of gated commits, which can prevent developers from breaking sources in a version control system. This is done by running the build remotely for local changes, before committing it.

TeamCity has been gaining popularity over the past few years, offering a decent alternative to other CI tools in the market. And, if you’re interested in viewing your builds and tests as they happen, or want a free and powerful CI solution, there’s no doubt that TeamCity is worth checking out.

Codeship

Codeship has a different view on CI/CD, and offers a Hosted Continuous Integration as a Service. This tool was originally built to give a Continuous Integration platform for Rails developers; hosting their code on GitHub and deploying to Heroku. Due to its popularity and demand, the company expanded over the years to support other technologies as well.

Codeship has 2 different variations of the product, each with its pros and cons. Codeship Basic allows setting up a CI/CD process by connecting a repository through a web UI and turnkey deployments. It offers support for a pre-configured CI environment, and allows multiple different builds to run on the same build VM.

Codeship Pro uses Docker to define a CI/CD environment, through it you can run your build pipeline. It comes with complete control over the build environment, letting you de-

fine what to run within it. The Pro version also allows pre branch caching, to set which image and which part of the workflow gets cached, along with parallel deployments.

As a whole, Codeship supports a number of languages, such as Java, Go, Node.js, Python, Ruby and others. When it comes to deployment, the Basic version supports AWS, Heroku, Azure and Kubernetes while the Pro also supports AWS ElasticBeanstalk, Google App Engine and DigitalOcean.

There is a public collection of utilities, scripts and Docker images to use with the tool, and the company even points out that some of these can be used with other similar tools. This collection includes deployment scripts for external services that are customizable, scripts to install specific versions of software that are not included by default on the build VMs and more.

Offering 2 different tools under one domain might seem a bit odd, but it gives Codeship the option to focus on various elements that will better suit different kinds of customers. Since both the Basic and Pro are available for free, it's an interesting choice for you CI needs.

GitLab CI

Soon after GitLab was launched, the team launched GitLab CI, the continuous integration service. Along with testing and building projects, the tool can also deploy the builds to your infrastructure, giving you an option to track your different deployments by knowing where each piece of code went.

GitLab CI is offered for free for as a part of GitLab, and can be set up rather quickly. To start using GitLab CI, you first need to add a .gitlab-ci.yml file to the root directory of your repository, as well as configure your GitLab project to use a Runner. Afterwards, every commit or push will trigger the CI pipeline that has three stages: build, test and deploy.

Each build can be divided into multiple jobs, and can run in parallel on multiple machines. The tool gives immediate feedback on the success or failure of the build, letting users know whether something went wrong or if something broke in the process.

GitLab (and GitLab CI) is an open source project. In other words, you have access and the ability to modify the GitLab Community Edition, and the Enterprise Edition source code. Plus, if you're using GitLab, it's almost a no-brainer to try the GitLab CI solution as part of the pack.

Bamboo

Bamboo is a part of the Atlassian product suite, and similarly to other tools, it offers building, testing and deploying code and supports numerous languages. It has strong integrations to other Atlassian products that are relevant for the CI cycle, such as JIRA and Bitbucket.

Build, test and deploy are all part of Bamboo's package, and the testing part is done with the help of a Bamboo Agents. Similar to the agents in Java monitoring, Bamboo also offers two types; the local agents run as part of the Bamboo server as part of its process, while the remote agents run on other servers and computers. Each agent is assigned to the builds that match its capabilities, which allows assigning different agents to different builds.

The main advantage Bamboo offers is the strong ties to the rest of Atlassian's products, such as JIRA and Bitbucket. With Bamboo you can see code changes and JIRA issues that have been introduced into the code since the last deployment. That way, developers can sync their workflow and always stay on track and know which version is next, and what (should) have been fixed.

Bamboo is a powerful solution for those using it with Bitbucket and JIRA, and are willing to pay for a CI solution.

Chapter 4

Container Orchestration Tools

Building and maintaining containerized software is entirely different venture than it is for monolithic apps, so it makes sense that container orchestration tools have sprouted and adapted to support this relatively new area of the software delivery supply chain

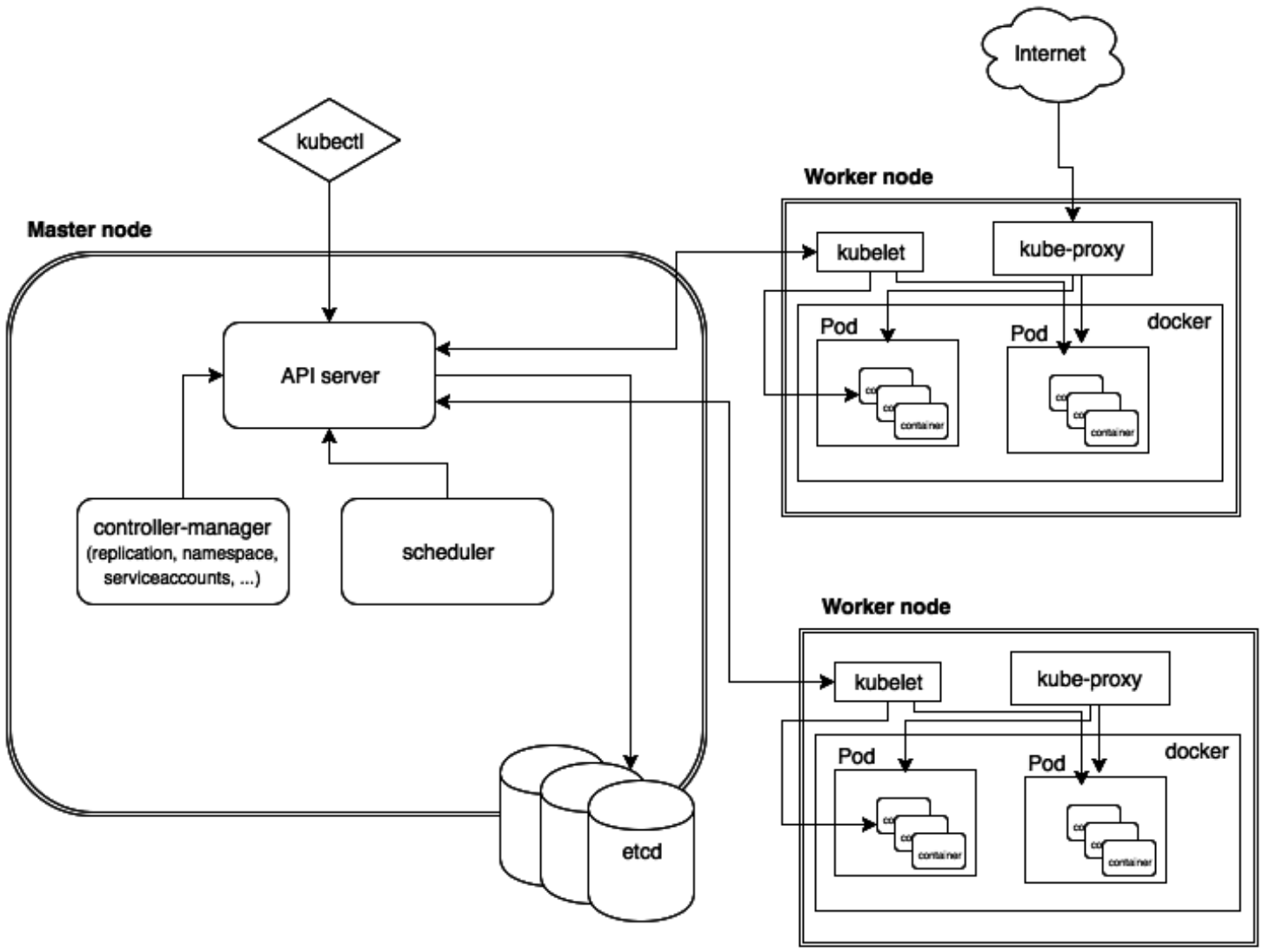
Container orchestration tools help to integrate and manage containerized software at scale. Each approaches the deployment of containerized, cloud-native applications differently than the others.

While there are many tools to choose from in this space (i.e. Azure, Docker, Mesos and more), we'll focus here on a comparison of 2 specific tools, how they approach container orchestration and why you don't have to choose just one tool to suit your application's needs.

Kubernetes

Kubernetes is a container scheduler or orchestrator. With container orchestration tools, the user creates and maintains the container themselves. For many teams, having this flexibility and control over the application is preferred.

Instead of focusing only on the app, the developer needs to create the container and then maintain it in the future, for example, when anything on the stack has an update (a new JVM version, etc).



(Source: x-team.com) Kubernetes architecture

Features

Kubernetes is a mature container orchestrator that runs in the same market as Docker Swarm and Apache Mesos. In Kubernetes, containers are grouped together into pods based on logical dependencies which can then be easily scaled at runtime.

More basic features of Kubernetes include:

- Master node for global control (scheduling, API server, data center)
- Worker nodes (VM or physical machine) with services needed to run con tainer pods
- Auto-scaling of containers and volume management
- Flexible architecture with replaceable components and 3rd party plugins
- Stateful persistence layer
- Kubectl (Kubernetes command line interface)
- Active OSS community

Installation and Usability

A common complaint that users have about Kubernetes is the difficulty of the setup process. First of all, you have to plan ahead when starting to implement K8s because you have to define your nodes in advanced, which can be very time consuming. On top of that, setting up Kubernetes varies for each OS, and the documentation isn't sufficient in cases when users need to build custom environments. Add to all of that manual integrations that are required, and even thinking about going through the setup process can give you a headache.

Once it's ready to use, though, Kubernetes offers the most mature and most popular service on the market in terms of container orchestration tools. It also has an active community offering support and resources to users.

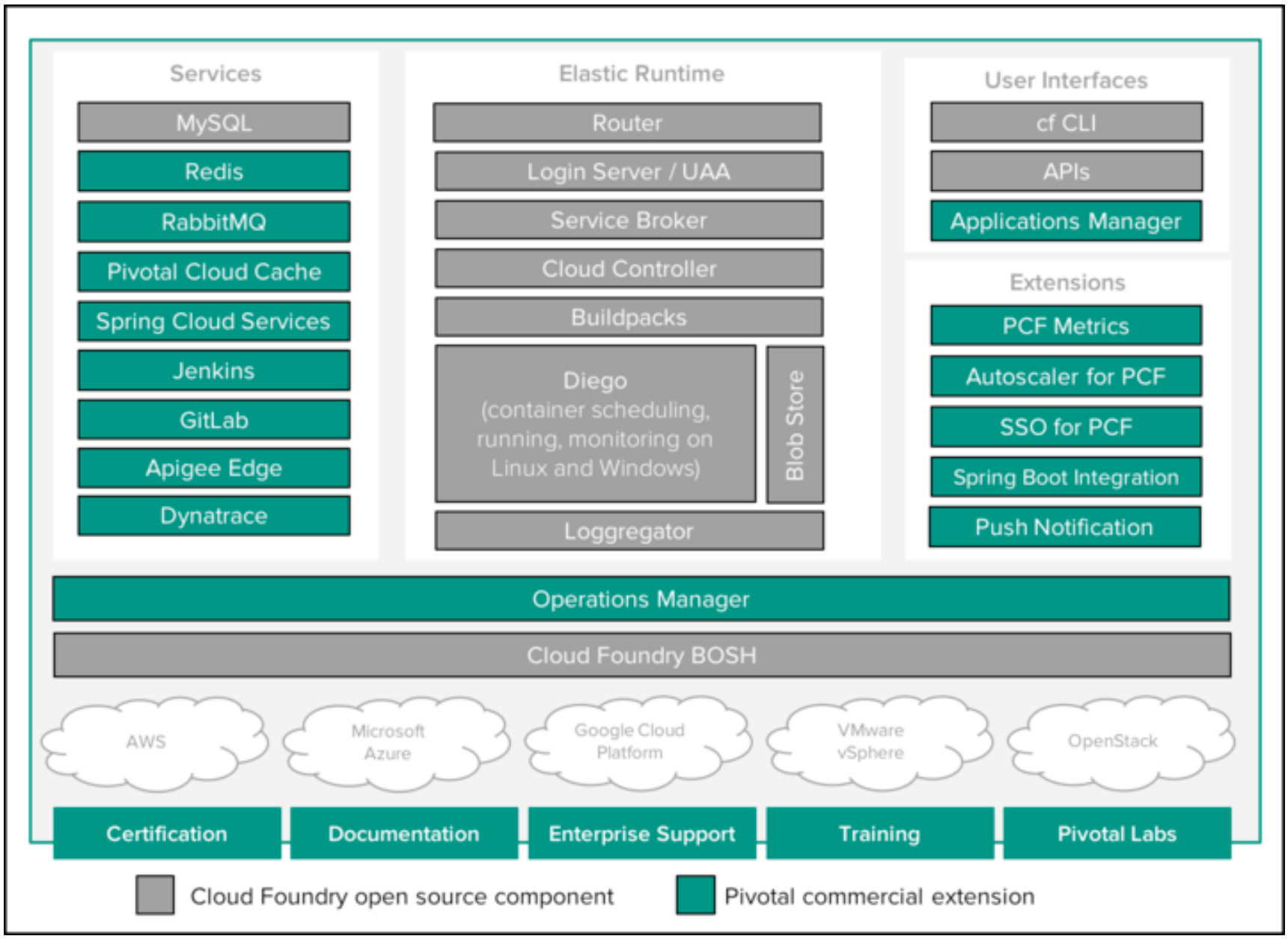
Best Use Cases

Kubernetes is a lower-level abstraction in the PaaS world meaning greater flexibility to implement customizations and build your containers to run how you want them to run. Unfortunately, this also means more work for your engineering teams and de-creased productivity.

When moving to any new system or product, a good rule of thumb is to use the high-est level abstraction that will solve your problem without putting any unnecessary limi-tations on the workload. If you need more flexibility to do customizations, and you're willing to put in the work, stick with Kubernetes (or check out Kubo below).

Pivotal Cloud Foundry (PCF)

Pivotal Cloud Foundry is a high-level abstraction of cloud-native application development. You give PCF an application, and the platform does the rest. It does everything from understanding application dependencies to container building and scaling and wiring up networking and routing.



(Source: pivotal.io) Pivotal Cloud Foundry architecture - open source and enterprise

Features

Applications run on Cloud Foundry are deployed, scaled and maintained by BOSH (PCF's infrastructure management component). It deploys versioned software and the VM for it to run on and then monitors the application after deployment. Although the learning curve for BOSH is considered to be fairly high, once mastered it adds considerable value by boosting team productivity.

More basic features of Pivotal Cloud Foundry include:

- Cloud Controller to direct application deployment
- Deploy using Docker Images and Buildpacks
- Automated routing of all incoming traffic to appropriate component
- Instant (vertical or horizontal) application scaling
- Cf CLI (PCF command line interface)
- Cluster scheduler
- Load balancer and DNS
- “Loggregator” – Logging and metrics aggregation

Installation and Usability

Before beginning the installation process for Pivotal Cloud Foundry, Pivotal documentation directs users to configure their firewall for PCF and establish IaaS user role guidelines. After that, installation is guided by a web user interface.

Best Use Cases

Cloud Foundry’s platform is a higher-level abstraction and so it offers a higher level of productivity to its users. With productivity, though, comes certain limitations in what can be customized in the runtime.

PCF is ideal for new applications, cloud-native apps and apps that run fine out of a buildpack. For teams working with short lifecycles and frequent releases, PCF offers an excellent product.

Best of Both Worlds: Cloud Foundry Container Runtime

The Cloud Foundry Container Runtime (CFCR), previously called Kubo, basically takes Kubernetes and runs it on top of BOSH, Cloud Foundry’s open-source lifecycle management tool. The goal of this collaboration between Pivotal and Google, the creator of Kubernetes, is to create “a uniform way to instantiate, deploy, and manage highly available Kubernetes clusters.”

CFCR gives users the customization abilities of Kubernetes with the deployment and management power of BOSH.

OR, Explore These Alternative Solutions...

1. Azure Container Service (ACS)

2. Docker Swarm

3. Mesos

Chapter 5

Log Management and Performance Monitoring Tools

Log aggregator tools and APM (Application Performance Monitoring) tools are two substantial pillars of the application troubleshooting tooling ecosystem

Log management tools aren't *just* log parsing machines, they're powerful data aggregators that can index massive amounts of data to bring out meaningful insights. The tools in this space can be used to get a better understanding and more value out of your logs, but they also have their own strengths and weaknesses. In order to get the most out of your tooling, it's important to select it based on your needs and goals.

Like with log management, APM tools try to make sense of what's going on inside your application. The difference is that rather than diving into the logs to find their information, they track performance metrics like uptime, throughput, etc. to alert you when something isn't work as it should. Again, there are some big names in this space, and choosing the right tool for your team is crucial.

Log Management Tools

Splunk

Splunk is the biggest tool in the log management space. It's well-established, full-featured, and enterprise-class. It's unique in this space as an on-premises tool (although they have come out with a Cloud version as well).

It's the most feature-rich option with more than 500 individual apps wrapped into the one package. It comes with built-in options for search and visualization and is a strong choice for security, BI and infrastructure monitoring as well as basic application monitoring.

As can be expected with an enterprise-focused tool, Splunk comes with a hefty price tag for those planning to use it to support real-world applications. It also has a slightly complex setup and demands a lot of maintenance to keep it running.

Elastic

Elastic (formerly ELK – ElasticSearch, Logstash, Kibana) is an open source project made up of many different tools for application data analysis and visualization. Logstash, specifically, was made for the collection and management of log files. Beyond log aggregation, it includes ElasticSearch for indexing and searching through data and Kibana for charting and visualizing data. Together they form a powerful log management solution.

Elastic is a good option for those wanting to have more control over their setup. Plus, it combines the functionality of 3 mature components into 1 powerful solution, and has a relatively easy setup for an open source project. One downside is that each component needs to be treated more or less as individual products because Logstash filters are written in Ruby, Kibana is pure Javascript, and ElasticSearch has its own REST API as well as JSON templates.

Sumo Logic

Sumo Logic was founded as a SaaS version of Splunk, going so far as to imitate some of Splunk's features and visuals early on. Since then, Sumo Logic has developed into a full-fledged enterprise-class log management solution in its own right. Sumo Logic is the most enterprise-focused of the cloud-native log analyzers.

This is a good tool for enterprise-type companies willing to sacrifice some features for the benefits of SaaS. It's also good if you have a strong focus on security. It's not just developer-oriented as a tool either, with benefits for security teams and business purposes. Sumo Logic provides advanced analytics and machine learning for logs, metrics and external data and tracks baseline metrics for anomaly detection purposes.

Like Splunk, Sumo Logic also does their pricing in 3 tiers. Though here it's based more on feature availability than it is on SaaS versus On-Prem. The free version covers you for up to 500MB/day, but that's not likely to cover you. For the Professional tier, you're looking at a starting price of $270/month for up to 3GB/day compared to $450/month for the Enterprise tier.

Plus, check out these other Log Management Tools:

- Loggly
- PaperTrail
- Graylog

Application Performance Monitoring (APM) Tools

AppDynamics

AppDynamics is undoubtedly one of the APM giants, serving mainly large enterprise companies. As a part of their "End-User Monitoring", AppD uses 3 different products: "Browser Real-User", "Browser Synthetic" and "Mobile Real-Time" monitoring. In simple terms, you'll be able to monitor user experience and interactions and benchmark your performance from specific regions.

It also captures business transaction anomalies with complete stack traces. Similar in a lot of respect to Dynatrace but perhaps more intuitive without the need to frequently drill-down/rotate views. The rest of the available information is automatically bubbled up to you, so it's visually easier to understand and act on.

Dynatrace

Dynatrace offers stiff competition to AppD with comparable features for End-User Experience Monitoring. They divided this into 3 products as well: "User Experience Man-

agement", "Synthetic Monitoring" and "Data Center RUM (Real User Monitoring)". The capabilities here come out more or less the same, the idea is to track metrics related to real user experience in real-time.

As mentioned above, Dynatrace also captures all user requests with limited stack traces (that include only the topmost elements). These profiles are known as "PurePaths" which visualize the journey of every request. Transaction profiling has always been a strength of Dynatrace, and they were one of the first vendors to trace transactions across heterogeneous run-time environments (e.g. hybrid JVM and CLR architectures). Dynatrace provides multiple drill-down options from a given PurePath so it's possible to slice and dice data from different angles.

New Relic

New Relic is most commonly thought of as the champion of SaaS-based APM and is one of the leaders of the space up there with AppD and Dynatrace.

The real bread and butter of New Relic's offering is backend monitoring including high level metrics with drill downs to code level data about how your application is performing. Must have metrics include transaction response time, error rate and throughput (Requests per Minute). Plus, of course, server and database monitoring with their accompanying metrics and insights like CPU usage, memory utilization, etc.

Aside from backend monitoring, New Relic also gets into End-User monitoring comparative to its competitors.

Pricing depends on instance size, runtime and quantity but it's easy to check for a price estimation with their pricing calculator on their pricing page.

Plus, check out these open source APMs:

- Stagemonitor
- Pinpoint
- MoSKito
- Glowroot

Chapter 6

Alerting and Visualization Tools

It's not over until it's over... And it's never over

CI/CD is taking us to exciting places, but having all the automation and management in the world can only take you so far if you don't maintain your production environment.

Alerting and visualization tools provide insight into the workings and status of your app in production. The biggest job for these tools is to connect the data that is coming from the rest of your suite of monitoring tools. After that, they use the information to provide different services aimed at making sense of the noise, whether that means pinging someone when a certain event occurs or revealing the overall trend of a unique event.

Although each tool in this list has its own upsides and downsides, it's important to remember that each one can only ever be as good as the data that it receives.

Alerting Tools

Pagerduty

Pagerduty is an alerting tool that pings you when issues arise in the different monitoring tools that you're having it watch. It doesn't monitor anything on it's own, but takes alerts generated by other tools and sends them out to you and your team based on escalation and priority rules you set up. It can send out alerts through email, phones, and several other means of contact. You can use it to collect alerts from your monitoring tools and create schedules to coordinate your team.

As the main player in this field with only a few competitors (such as VictorOps), this is the tool to check out for those looking for one unified alerting solution. Also give it a try if an inside-your-environment alerting tool is what you're looking for in general.

To get the most out of it, you have to spend a bit of time hooking it up with your different tools and entering in the rules for the different alerts you want it to give. It's also one of the most expensive tools in this area. Pricing for Pagerduty ranges from $9-$99 per month per user, depending on feature needs.

VictorOps

VictorOps is an alerting tool, or more perhaps more accurately, an incident management tool that was built specifically for DevOps. One of their most celebrated value-propositions is helping to streamline the on-call experience in incident management situations.

Some features which make the distinction between an alerting and an incident management tool are:

- The "Transmorgifier" which allows teams to add contextual information to the full incident timeline
- The bi-directional chat in-app and with integration to Slack and HipChat
- Integration with GitHub/Jenkins for extensive SDLC insights, and post-incident documentation.

Following its acquisition, VictorOps' routing and more is sure to get a boost from Splunk's capabilities in and around artificial intelligence and data aggregation.

Pingdom

Pingdom is a service that provides tracking and alerting on website's response times, 24/7. It helps answer a crucial question that may seem trivial at first blush: Is your website available? By probing it from different locations all over the globe, it can help differentiate actual downtime from routing and access problems.

This also means, though, that it's only usable for external-facing apps and environments. Plus, the level of sophistication for the alerts is fairly low. Still, for those looking for an outside-your-environment alerting tool with quick and easy set up, this can be a good option.

Visualization Tools

Graphite

Graphite is an excellent open source tool for handling visualizations and metrics. It has a powerful querying API and a fairly feature-rich setup. In fact, the Graphite metric protocol is often chosen the de facto format for many metrics gatherers.

However, it isn't always a straightforward tool to deploy. Part of the reason for the deployment pains is that it's made up of three distinct elements (well, it requires four if you include the metric gatherer), and depending on your environment, one or more of the default elements may not be satisfactory for what you need.

While having three components can cause some implementation headaches, there's a positive result as well. Each piece is a distinct unit, so you can mix and match which of the three elements you actually use. That means you can build a fully customized Graphite deployment just for you.

Aside from the Metrics Gatherer (CollectD, StatsD, etc.) which isn't actually part of Graphite, the components needed to run Graphite are: The Listener to report incoming metrics to the database, The Storage Database to collect the metrics, and The Visualizer (Graphite-Web) to view them.

Since the components operate individually, it's not uncommon for users to replace the visual component with the more refined and, well, pretty Grafana dashboard.

Grafana

Grafana is an open source dashboard tool that works with both Graphite and InfluxDB. It used to be a front-end only tool that required ElasticSearch for storing dashboards, but it now comes with a backend storage component for storing dashboards you create. It was originally designed to create a better visualization component for Graphite, making it a strong contender to replace the default Graphite-web.

Also check out...

- Kibana as a part of Elastic
- DataDog
- Keen IO
- Liberato

Final Thoughts

Supercharge Your Supply Chain with OverOps

The changes over the past few years in the software delivery supply chain are phenomenal. There is an almost entirely new set of tools that we use now to create, deliver and manage our (software) products. This includes both open source and proprietary tools, all of which can be implemented in combinations to create a successful delivery model.

With access to all of these tools that help us constantly accelerate software delivery, it's important to remember the importance of ensuring Continuous Reliability of our products. The bottom line of every tool that promotes product success is data. Without improved data, these tools are less capable of keeping up with the fast pace of software delivery.

OverOps reinvents how we gain code-aware insight about errors and exceptions in our software. This new approach provides deep insight into application reliability in any environment, including production. With the recent release of the OverOps Platform, we're building bridges between the data we collect and the rest of your tooling ecosystem.

Some of the key data we provide that can be leveraged by the rest of the tools in the supply chain are:

- State of the JVM at time of error, including:
    - Full Stack and Variable State
    - Source Code
    - JVM Memory State

- 250 log statements leading up to the time of error
- Deployment/Build/Release Information
- AND MORE

Building the perfect enterprise DevOps stack, or really building any kind of tooling ecosystem, is about looking at the problem at hand and understanding what ties everything together. When it comes to building and maintaining reliable software, data is everything.

Learn how you can supercharge your current workflow with new data from OverOps here.