Quo Vadis? PaaS, Enterprise IaaS or Cloud Native?

DRAFT DRAFT DRAFT

The inspiration for this article is a panel, I am going to join in 2 weeks. The panel discussion gather experts with many years of Microsoft experience and most of them works for Enterprises. Me, an Englishman in New York, I mean Redmond.

Let's take a look on the options, we have for the enterprises or any company in the end. Let's define what the options we have on PaaS, IaaS, or Cloud Native.

Summary

The Kubernetes / CloudNative is widely accepted as the next step for the infrastructure. The history of using Kubernetes is at early as 2016 in insurance and bank companies in Poland.

The IaaS is here to stay to keep your core services running. Whether you need new features, you put them as services on top of K8S.

PaaS lost its steam. Its legacy lives in CPaaS. CPaaS lets you standarize your development process in PaaS spirit, but on top of kubernetes. It gives you containers when you need flexibility and templates with runtimes when you need standard. In 2018, Pivotal released CloudFoundary on top of K8S. Suse, gives you CPaaS with CloudFoundry runtimes

The vendor-specific PaaS in the cloud is here to stay. Most prominent - Microsoft Azure and Google App Engine. With the rise of CPaaS, such as OpenShift, PaaS will move in the direction of a plugin/extension platform for the SaaS products. It happened to Heroku, the initial idea of SAP Cloud was exactly about it. Here it will naturally crash with Serverless in future.

Coming back to Kubernetes, the project ambition is to become a framework platform. So, anybody can build their own Xubernetes. However it is not clear whether, we will see such a diversity as Linux or Openstack few year ago. So more interesting development ahead of us.

For most companies, the most relevant question now is for many companies OnPrem or OnCloud. Experiencing growing pains of adapting internal processes to Kubernetes.

Another group are companies that worries about the security properties of containers. We have many companies working on bringing virtual machines under kubernetes, so the last obstacles should be removed in 2019.

Let's see which options we have as a company.

CaaS, Cloud Native, and Kubernetes

The containers management platform won, the trend started by Mesophsere, and now Kubernetes.

Container as a service might not be valid term, because we have many projects working making Kubernetes manage virtual machines. We have: Kubevirt.io (https://kubevirt.io/), virtlet (https://github.com/Mirantis/virtlet), [1], or Amazon Firecracker [[2]].

For me, if my app is run by Docker or in a micro-vm, does not matter as long as the startup time is fast and I do not have any issue to run them on my development machine.

CaaS Managed onCloud

Here are our leaders in managed CaaS:

1. Google Kubernetes Engine
2. Azure AKS
3. AWS EKS
4. AWS Fargate

Difference between them:

1) GCP most mature and most stable
2) Azure AKS, longer in the game, still issues with stability
3) Kubernetes at AWS, you have the cloud provider with 78% market share and longest history of providing cloud at scale. All the XXX of the other services.

Choose your medicine.

My take. New to cloud. Just want to run K8S - GKE. Although, especially in USA, companies are concerned that it is not the main business of Google.

You want K8S, VMs, and unknown-unknowns-you-might-need - AWS. Azure Kubernetes Service is also a solid option. Especially, when you are in the MS ecosystem and want to leverage, .e.g., licensing [3].

CaaS Self-Managed

The most of Kubernetes installations are installed by the companies themselves. You will find

The usual suspects - Ubuntu and CentOS/Redhat:

[1]https://github.com/kata-containers
[2]https://firecracker-microvm.github.io/
[3]https://azure.microsoft.com/en-us/pricing/hybrid-benefit/

1. Redhat OC or OpenShift
2. Ubuntu Kubernetes
3. Rancher 2.0
4. SUSE CaaS / SUSE Cloud Application Platform [[4]]
5. GKE-on-prem
6. Docker on Windows

Many more: [5]. I have omitted Mesos, Docker Swarm, or Nomad, because they are less popular. Recently Mesos positions itself as a tool to manage your kubernetes platform. Something that Rachner does extermly well.

I would much more prefer AKS-on-Prem on mixed nodes Ubuntu and Windows and not getting Docker-on Windows. Docker-on-Windows together with Enterprise IaaS might be a good way to run your legacy. New developments should go on CaaS.

Although, in many cases, it is better to leave it on your IaaS (most probably it is VMWare or Windows), and use money and time to push new software.

The companies are learning that "can I run it in the container?" and "does it run on Linux?" should be a part of any procurement process for new software.

To research:

- what the secure opinionated K8S is? What the default network plugin for RH, SUse,UBubtu, Mirantis… or other provider is???

PaaS, APaaS, and CPaaS

I love Heroku experience, although I have never used it for any heavy lifting. The deployment model inspired the coming products for years. Pivotal, few years back, took everybody by surprise with their Cloud Foundry.

The rise of the containers, Kubernetes in particular, the novel approach of RedHat with OpenShift, pushed the market to introduce two new terms: Application PaaS and Container PaaS (CPaaS) - Openshift in particular. At this stage, many companies go for the pure Container-as-a-Service solution. The most popular is, of course, Kubernetes.

Application PaaS

onCloud

Let's take a look what the options we have here:

Managed onCloud:

1) Google App Engine
2) Microsoft Azure PaaS

[4]https://www.suse.com/products/cloud-application-platform/cloud-foundry/
[5]https://kubernetes.io/docs/setup/pick-right-solution/

3) Cloud Foundry-based, e.g., SAP Cloud
4) Heroku

The rise of the significance of Kubernetes and the fact that k8s become the common runtime, we see that the k8s become the default runtime to deploying PaaS by providers:

1) IBM BlueMix Cloud Foundry
2) Pivotal Cloud Foundry Stack on (PAS on PKS) on Google Kubernetes Engine (see, [6], [7])

The application PaaS, you look on the SAP, Salesforce, or Microsoft materials, it becomes a plugin platform for your core IT system. In case of SAP, the SAP cloud let's us to extend our SAP ERP or Successfactors with custom implementations. The same story for Heroku, Salesforce uses it as a extension platform for their SaaS product. With the rise of CaaS, this trend will accelerate.

It should not be a surprise that brought Redhat OC on Azure and IBM bought Redhat to get relevant in the new reality.

OnPrem

Self-managed onCloud:

1) Cloud Foundry on your favorite cloud provider
2) Suse Application Cloud Platform

Container PaaS

All the main players, have a Kubernetes at the heart of its solutions.

1) RedHat OpenShift [8]
2) Cloud Foundry with VMWare [9]
3) Suse Cloud Application Platform [10]

Each of the main players went in a little different diction. Redhat decided to dump CF long time ago, most probably, from commercial reason (getting itss possiont in the enterprise) and practical - CF was/is a complex solution. The Kubernetes helped Redhat to deliver an almost PaaS (moving the runtime injection to CD/CI pipeline as a basic Docerfiles inside containers). So, they managed to have both, a simplicity for the developer, calm sleep for the enterprises (you can audit what is deployed, restrict base images), and keep your operations simple.

Pivotal, on the other hand, double-downed on Cloud Foundry (CF), its successful PaaS product, with introducing K8S as CF runtime platform. They build new

6 https://www.loomsystems.com/blog/everything-you-need-to-know-about-pks-in-3-acts
7 https://www.cloudfoundry.org/wp-content/uploads/CFF-User-Survey-April-2018.pdf
8 https://assets.openshift.com/hubfs/pdfs/OpenShift_for_Developers_Red_Hat.pdf?hsLang=en-us
9 https://www.loomsystems.com/blog/everything-you-need-to-know-about-pks-in-3-acts
10 https://www.suse.com/products/cloud-application-platform/cloud-foundry/

CF with 2 layers: PKS and PAS. PKS is a pivotal Kubernetes deployed with the BOSH tool. PAS is our old good cloud foundry. Users report [11], that the strongest support you will get for VMWare VSphere and GKE.

If you want to have Cloud Foundry on not opinionated Kubernetes, you should check Suse Cloud Application Platform. Suse, an Opensoure Phoenix that rose from the enterprise ashes, has a very interesting proposition for you. You will get Cloud Foundry running on pure Kubernetes. So you do not have a vendor locking (with VSphere benefits and pains. It works the same way that Redhat OC, it moves your application to a docker and injects all Cloud Foundry magic as a base container. If you wonder how, you can check the fissile tool that converts your BOSH releases to docker images.

The same tool IBM uses to move existing IBM Bluemix Cloud Foundry applications to their Kubernetes-based environment.

The most of the contributions to Cloud Foundry still are from Pivotal (stackanalytics). The time will show whether the more container-based Cloud Foundry, the approach less best-run-only-on-VMWare and more dependent on generic Kubernetes runtime bring the second live to the project. We see more commits from Google, SAP still relays on Cloud Foundry for SAPCloud. Although, SAP joins the Kubernetes with projects, such as gartener and Kyma.

XaaS

There is a large number of ready to use solutions that you are going to mix with your CaaS. In most cases, you will use SQS, SNS, Firebase, and multiple other tools solving your problem.

We might see more and more plugins that will let you to manage your external tools.

The common use case:

1. CaaS on public
2. CDN, . . . , SNS on AWS
3. GoogleAnalytics on Google

The technologies that supports it, such as Terraform, will be o rise.

IaaS

IaaS is here to stay. Still the best way to start develop new projects. As for running your application, IaaS is where you keep your beloved pets.

Public IaaS

1. AWS - circa 80%

[11] https://www.reddit.com/r/openshift/comments/96rxhr/openshift_vs_pivotal_cloud_foundy/

2. Azure
3. GCP

Enterprise IaaS - onPremise

Ok. So we know what we have on our plate. So we can have it self-hosted or in our basement.

Options:

1. VMWare
2. MS XYZ
3. Redhat RDO Openstack
4. Ubuntu Openstack
5. Suse Openstack

It is interesting. RH, in many materials and publications, says you should not go for IaaS, but jump as soon as possible to the containers, in particular, RH Openshift.

Serverless

Serverless onCloud

1. AWS Lambda - Most mature and provide the larger number of options
2. Google Functions - Limited stable
3. Azure Functions - Still in progress, recently rewrote

Serverless onPrem

TBD: Technology products based on Kubernetes.

Mixed but limit the options

DRAFT DRAFT

As the micro-service trend shows you should take the technology that is the most suitable for you problem. Still, you should limit the number of the technologies. The minimum, what we need to keep, are the conventions that let's us hide the details under a common abstractions, e.g., single point of entry, the same documentation format, ready to use scrips in common languages, and use all the automation in CI to enforce the state in your system from your version control.

You might have in place chef or puppet, from the infrastructure-as-a-code wave from few years ago. So, you might already have CI or CD in place. It is the best way to manage, control your technical debt.

DRAFT

Hybrid Cloud

Notes on the stack:

1. CaaS for New
2. IaaS for Pets
3. XaaS for solving problems
4. Serverless to cut the costs

The tech:

1. Kubernetes
2. IaaS or Docker-for-Pets for your pets, .e.g., SAP ERP
3. XaaS
4. Optinal to cut costs

Directions:

1. Linux and linux containers

2. Bringing better tools:

- Terraform
- CRD in K8S

3. Bringing better processes:

- procurement
- audit
- management

4. Security:

- Google BeyondCorp

Annex

Context

The aim of this chapter is to see the context of the decisions.

How it fits in the landscape

Let's take a look what technology we usually have inside a companies.

Microsoft Windows is the leading desktop environment. Outlook still the most popular email client. Excel and MS Word dominates, although LibreOffice . LDAP, the most popular way to provide authentication and authorization.

Usually, the desktop is running tick or thin clients. Here we can find two leading technologies:

- Citrix-based thin clients, usually on VMWare or Citrix Xen
- and open-source alternative - guacamole - on anything

The Office365 is given to many customers to inflate the percent of Azure Cloud market share. You can find much less often Google Business or Zoho Suite.

Also — in case of bigger enterprises — a lot of licenses bought and not used.

Regarding the ERP, you will usually have SAP ERP, Oracle or MS Dynamics. interestingly SAP is committed both to PaaS (Cloud Foundry) and Kubernetes.

Keeping SAP as a core of your IT infrastructure, you extend it with addons running on kubernetes cluster to make your features delivered faster and using the modern software practices (@TODO REF missing). You can also mix onPrem and onCloud with SAPCloud with its PaaS based on Cloud Foundry.

I do not have so much connections to Microsoft and Oracle engineers (TODO: get review sb from AUGW). Regarding Microsoft, they follow the SAP suite, and also mention its onCloud PaaS as a plugin platform for your core systems.

Existing licenses

Challenges in moving the legacy / core to container or virtualized environment. There is sometimes often distrust in OpenSource, e.g., Postgres instead of Oracle.

Conditioning Decision Makers

- Digitalization, pressure to change
- Minimize risk, aka, if the project fails with big vendor X, it is not my fault. If the project fails with smaller vendor Y, I will loose my job.
- Need sth to show I am doing sth

Recruiting

- Hard, less people want to work for the corporations
- Who does want to work with MS Windows?
- Exciting: K8S, Cloud Native, Golang, .Net core2. . . . ML.

Procurement Processes

TODO: How the process fits to the procurement, controlling, vendor-locking, etc.

How to?

NOTES NOTES NOTES

Homework process:

1. Audyt, większość firm/startupów nie wie co ma. Nawet brakuje tagowania i grupowania maszyn.

2. Audyt: otagowanie, opisanie maszyn, żeby wiedzieć co tam jest i jaki produkt jest uruchomiony.

3. Audyt: otagowaliśmy, a to czego nie wiemy mówimy, że wyłączamy za tydzień, wyłączamy, jeśli było potrzebne zazwyczaj dowiadujemy się w kilka dni przed terminu wyłączenia.
4. Dużo firm nie ma rejestru zakupów i licencji, ani kalendarza z datami kiedy je odświeżyć. Często płacimy i nie używamy.
5. Dokumentacja - w dużych organizacjach - wiki - mediawiki albo confluence, żadnych wordów. Dokumentacja powinna obejmować aspekty operacyjne i aplikacyjne.
6. Najlepiej mieć jeden przykładowy process, dokumentacje, skrypty, aby inne teamy mogły klonować sprawdzone podejście.
7. Uwaga - 90% apek w enterprise nie ma zadnego API, trzeba najpiew przeleciec cala firme i zidentyfikowac co nadaje sie do automatyzacji, i pewnie najpierw zrobic plan co wywalic i zastapic.

Uwaga: CloundNative / … bez automatyzacji i pipelineów to po prostu przenoszenie bałaganu w inne miejsce.

NOTES NOTES NOTES.

Step 0. Implement small new project in YOUR TARGET TECH for your transformation.

Step 1. DevOps Culture, e.g., move all the traditional show stoppers to the left of the software development process. Test in the small project, the DevOps way.

Step 2. Homework. When moving existing applications (patrz wyżej ˆ).

0. Do not lift&shift immediately
1. Review existing applications, machines, servers
2. Detect and remove unused servers
3. Detect and remove unused licenses
4. Write in one place all the knowledge about component, automatize or semi-automatic the setup (e.g., as a screen-shot steps, README files)
5. Add smoke tests, the simplest ones that captures the notion whether the application is working
6. Migrate all the UI-clicking to code or scripts to the version control repository dedicated to given installation or your app.

If you have CI/CD in place, it will help a lot. You might, unfortunately have many applications installed by hand.

Step 3. Do the process in iterations.

Step 4. Add checklist-based approval process for new application/license purchases that takes into account how it fits to modern approach, e.g., whether it can run in container, at least on Linux, how it effects the license.

Not everything needs to be migrated. Very stable, not changing components, your core systems you might leave where they are and use PCaaS, CaaS, or PaaS to extend its functionality.

DRAFT DRAFT

Bibliography

DRAFT NOTES REFS TO USE

- Where we are with the cloud, https://www.zdnet.com/article/top-cloud-providers-2018-how-aws-microsoft-google-ibm-oracle-alibaba-stack-up/
- https://www.rightscale.com/lp/state-of-the-cloud
- https://turbonomic.com/