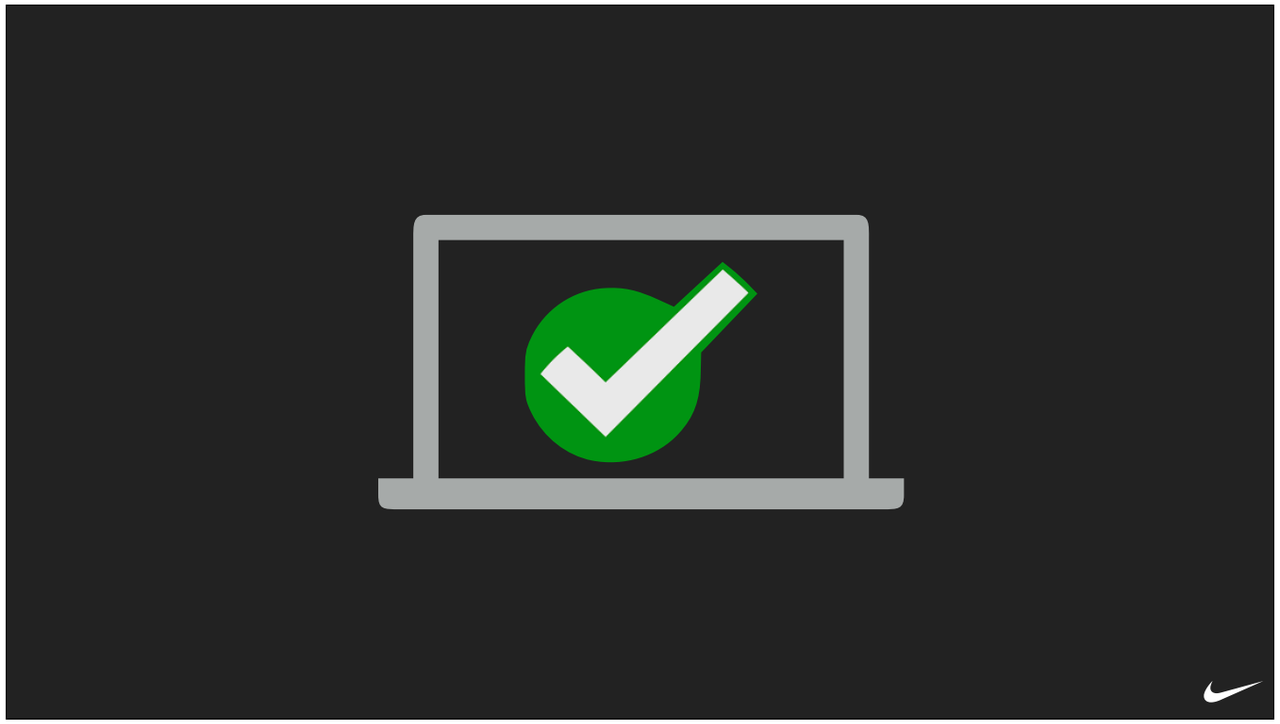
This is:
Get in the Game:
How to Use Referee and Kayenta to Do Canary Analysis



Imagine, you’re at work, sitting at your desk, and seems that all is well.
You just deployed a minor code change where you added additional unit tests. You glance at your production graphs and everything looks healthy, so you direct your attention to more urgent work.

A few hours later your phone starts buzzing. Your phone is blowing up!

Turns out your recent deploy has caused your service to go haywire.
Users are all receiving 500 errors and you have no idea why. You are floored, this was just a simple unit test and you are now facing a P1 incident. How in the world are you going to fix this?
How many of you have ever experienced anything like this?

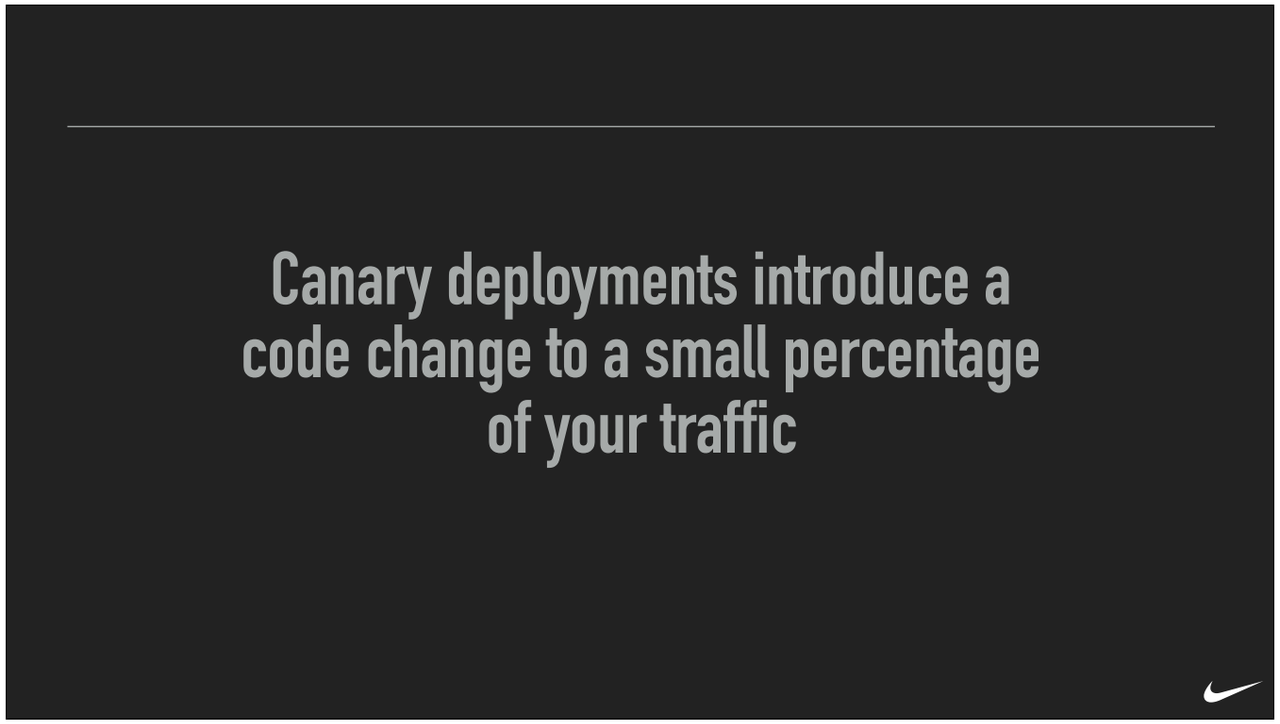
raise hand and wait for audience to raise hands

What did you do to solve it?
Maybe you dug into logs, reran your tests, crossed your fingers.

What if this wasn’t the way it had to be.
But what if there was another way...a better way...where would you never even get to this point to begin with...

My name is Melana Hammel and this is Justin Field.
We are both software engineers for Nike.
Nike’s ecommerce business is in the billions.
That means that for any given outage or production issue in one of our consumer facing services, there is a lot of money on the line.
We work on a platform team where we create cloud-native solutions to enable other engineers at Nike and encourage CI/CD practices.
We wanted to reduce the risk of deployments for the engineers that we serve.

Our search led us to what we are here today to share with you: CANARY DEPLOYMENTS.

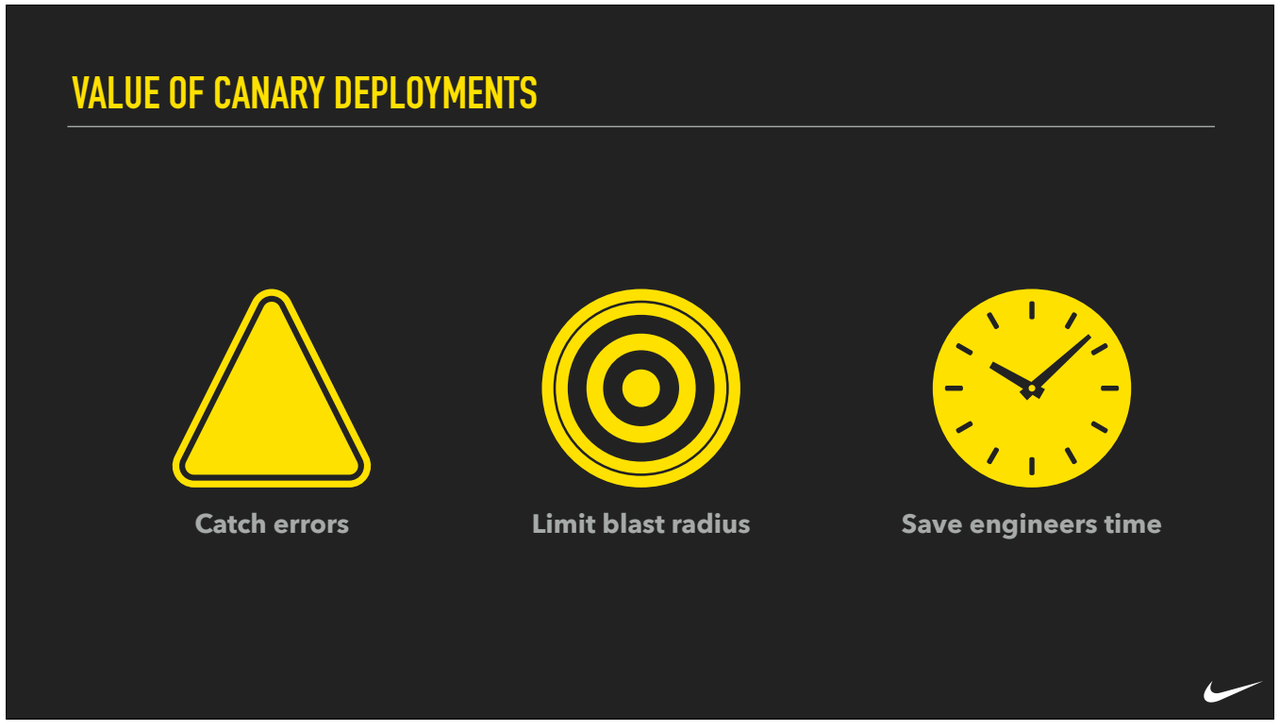


What are canary deployments?

Canary deployments introduce a code change to a small percentage of your traffic.
Where does the term canary come from?

Canary in a coal mine. The reference goes back to when miners brought canaries down into the mines with them. If the canary stopped singing, and died, the miners knew to get out of the mine because there was a carbon monoxide leak.

Canaries are known as an early indicator of potential failure or danger.

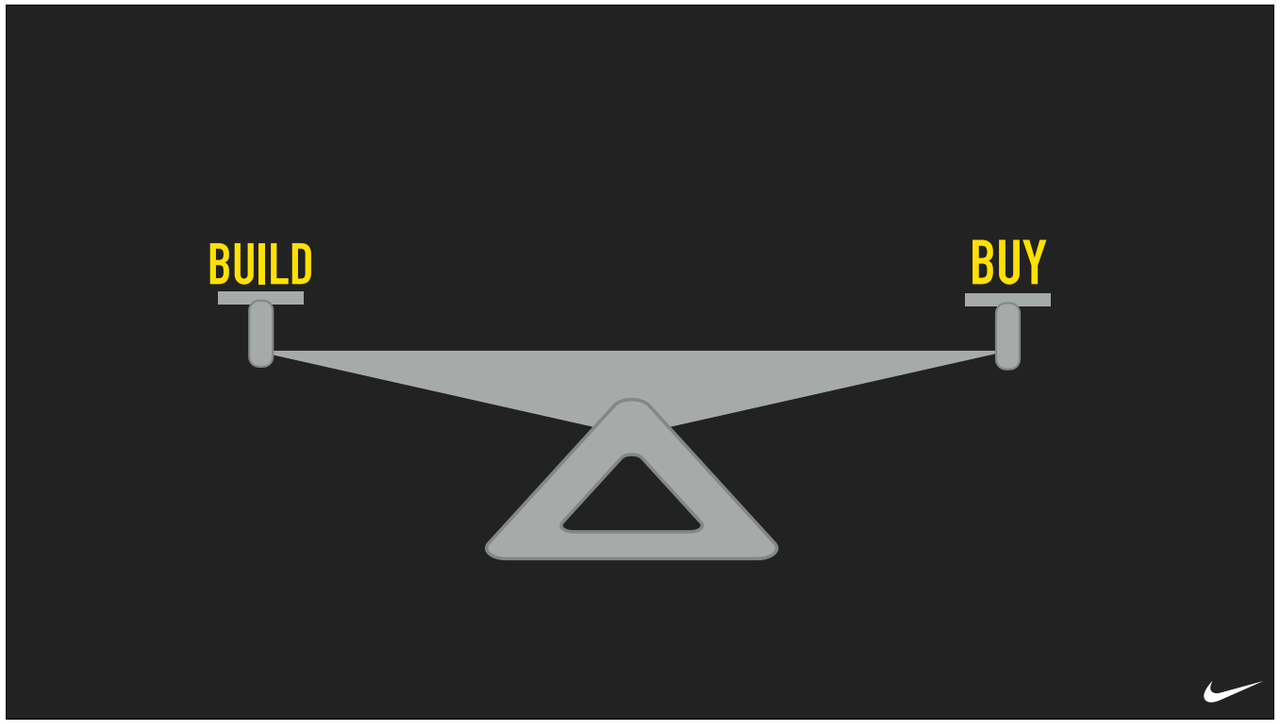


Canary deployments offer a lot of value.
Canaries can help catch mistakes and errors that would have previously been deployed directly to production.
Canary deployments offer a way to test new code in production while limiting the blast radius if anything goes wrong
Because canary deployments can be automated, they save engineers time because engineers don’t have to monitor logs or watch charts as often as they used to.

So we needed the ability to canary at Nike..
Now Justin will walk through our journey to find a canary solution.

We needed the ability to canary at Nike

We knew we wanted a first class Canary experience as part of our continuous delivery solution. In our search for a canary solution...

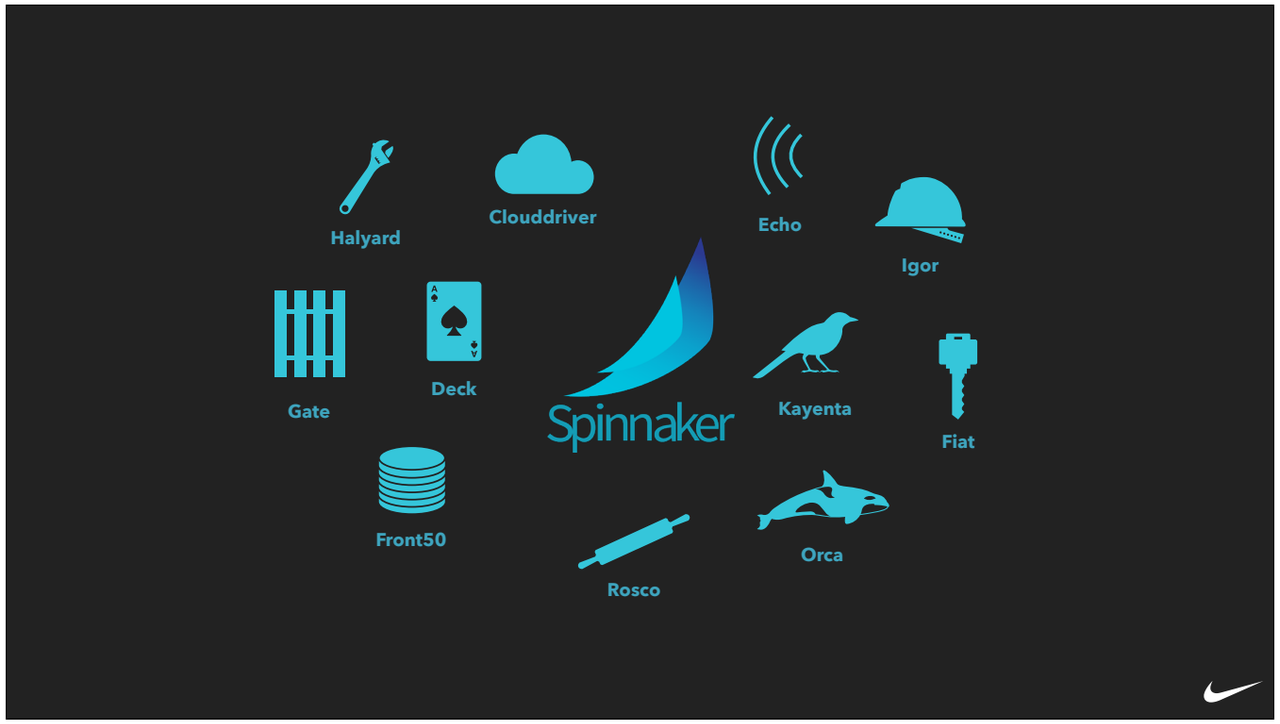


We faced the classic build vs buy scenario.

We had an internal solution but knew it needed major work to improve the user experience to be where it needed to be.

We could spend the time and energy to uplift that internal solution.

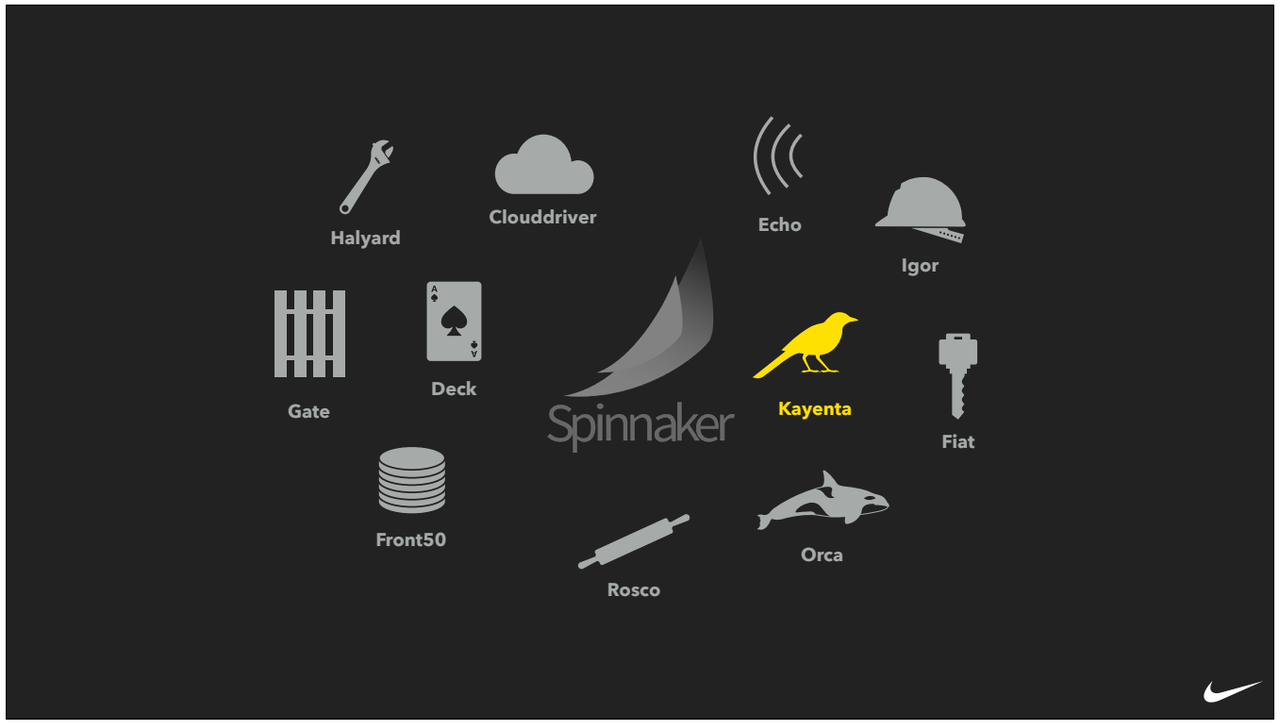
But because of our belief in open source software and the benefits of cross company collaboration, we looked into canary work in the open source space.



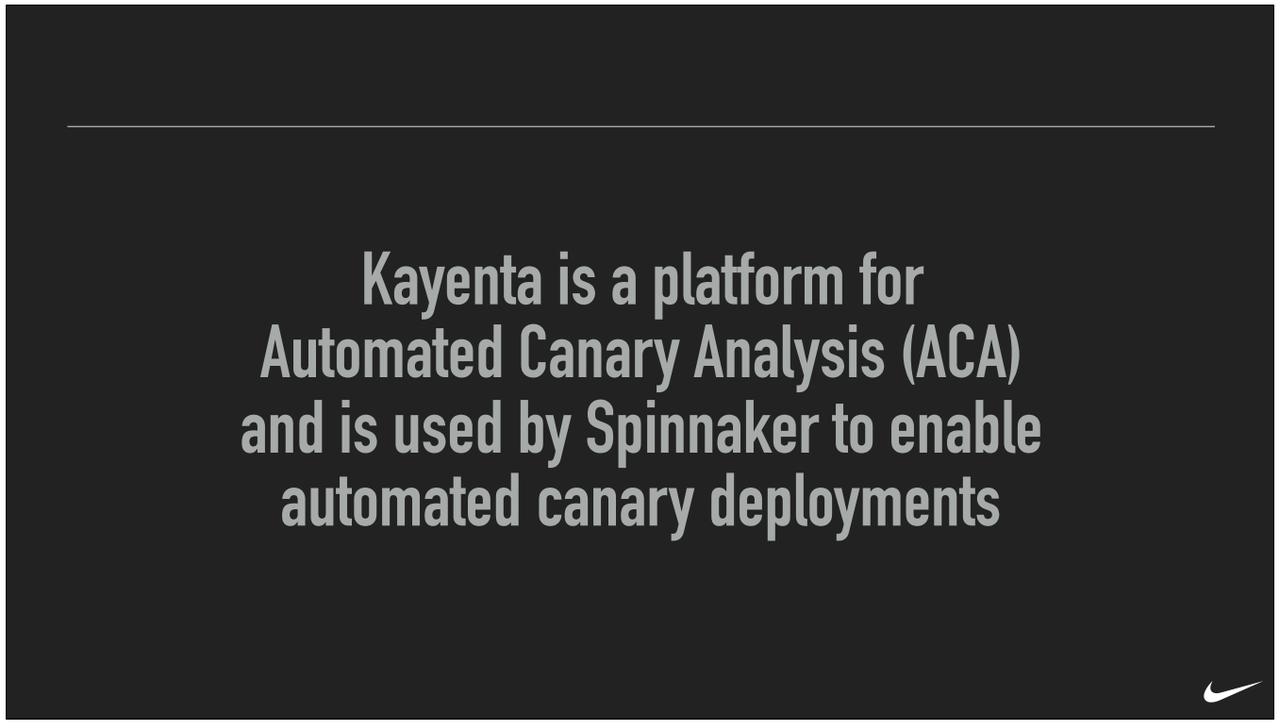
Our search led us to Spinnaker’s Kayenta

Spinnaker as you probably know by now is an open source, multi-cloud continuous delivery platform

Spinnaker is composed of many microservices that work together.



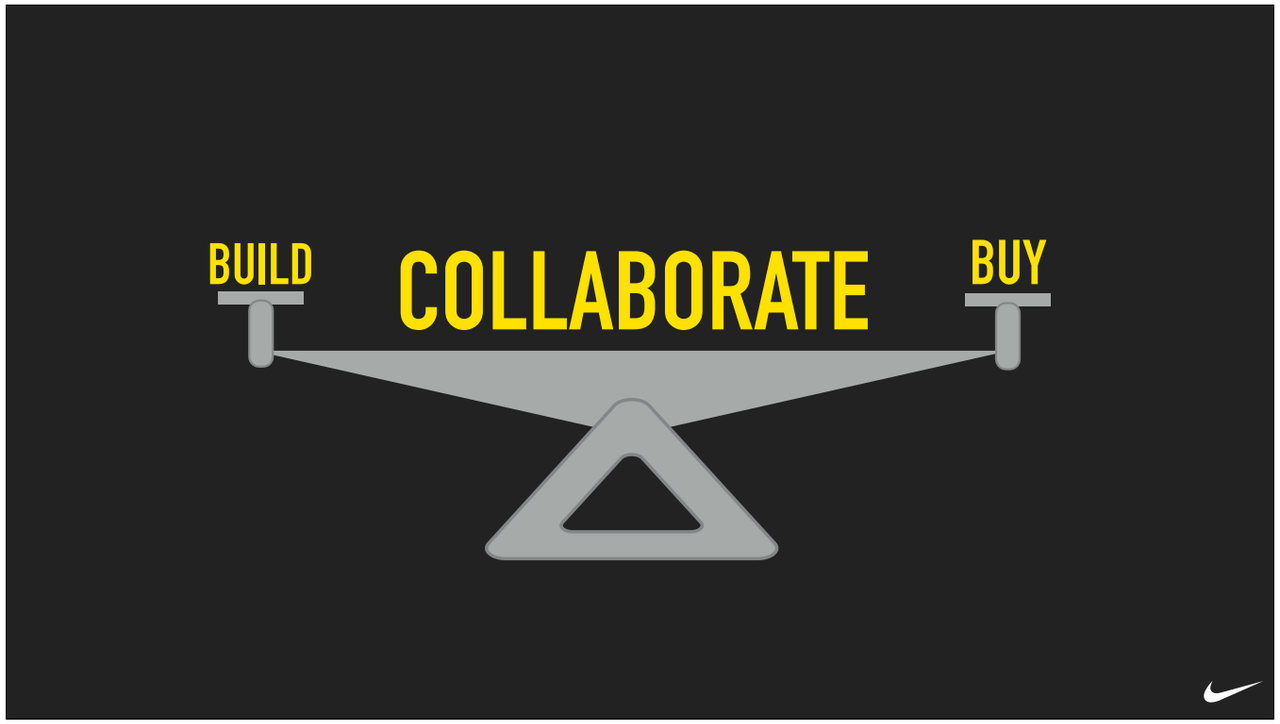
One of those microservices is Kayenta



Kayenta is a platform for Canary Analysis, it is capable of integrating into many metrics sources and doing a-b style statistical testing.

We identified it as an industry standard for canary analysis, having the benefit from the work of a lot of data scientists from multiple companies across the industry.

This was a great opportunity for us to align with the work of the OSS community.



So we found ourselves actually at a third option: **Collaborate**.

We couldn’t use the complete canary experience that Spinnaker as a whole offered, we just wanted to use Kayenta as a standalone service..

Kayenta did almost everything we wanted and we really liked the user experience that Spinnaker offers out of the box for Canary.

So we decided we wanted to extend Kayenta to meet our needs and contribute any work that needed to be done upstream so that others could do it too.

We were at last year’s Spinnaker Summit.

We had some early conversations with the Spinnaker Kayenta community about using Kayenta as a standalone service.

Now a year later we are presenting!

So what did we come here to tell you?

Canary. Use canary deployments when deploying changes to production.

Make your life easier as a developer.

Spend less time manually watching graphs and logs.

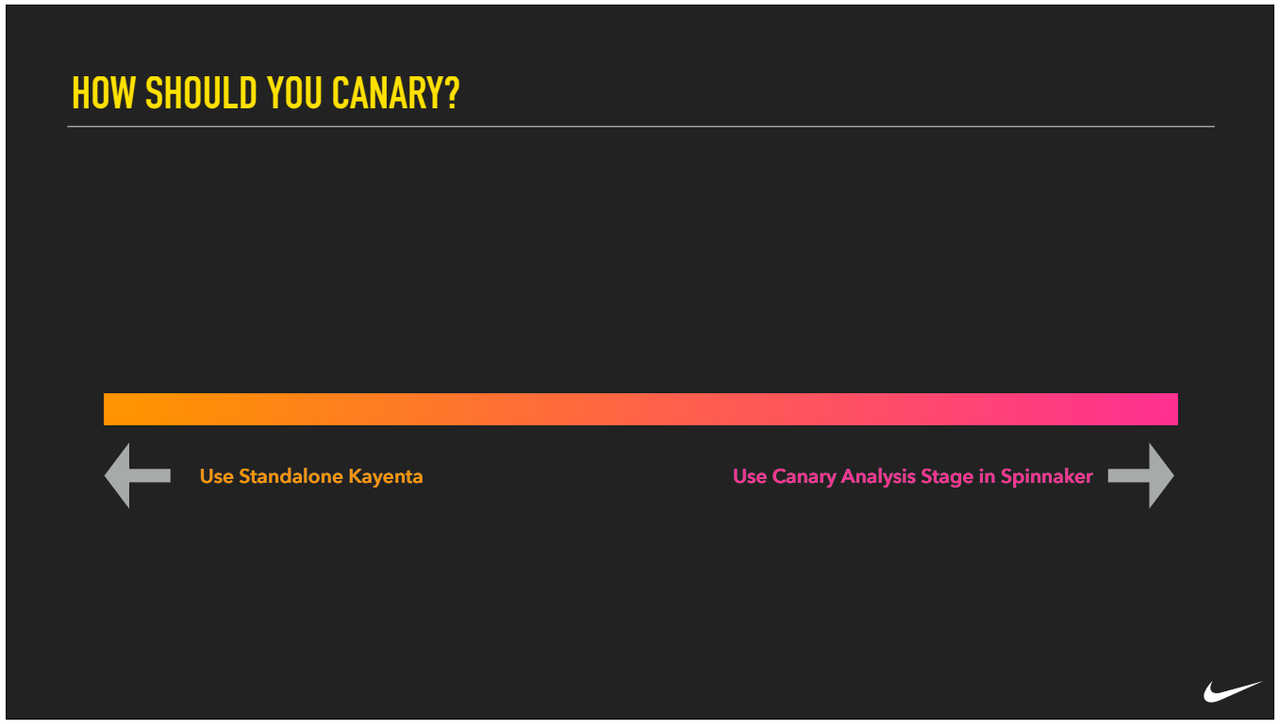
Prevent large scale production issues.

Use canary deployments!

You can canary two different ways.

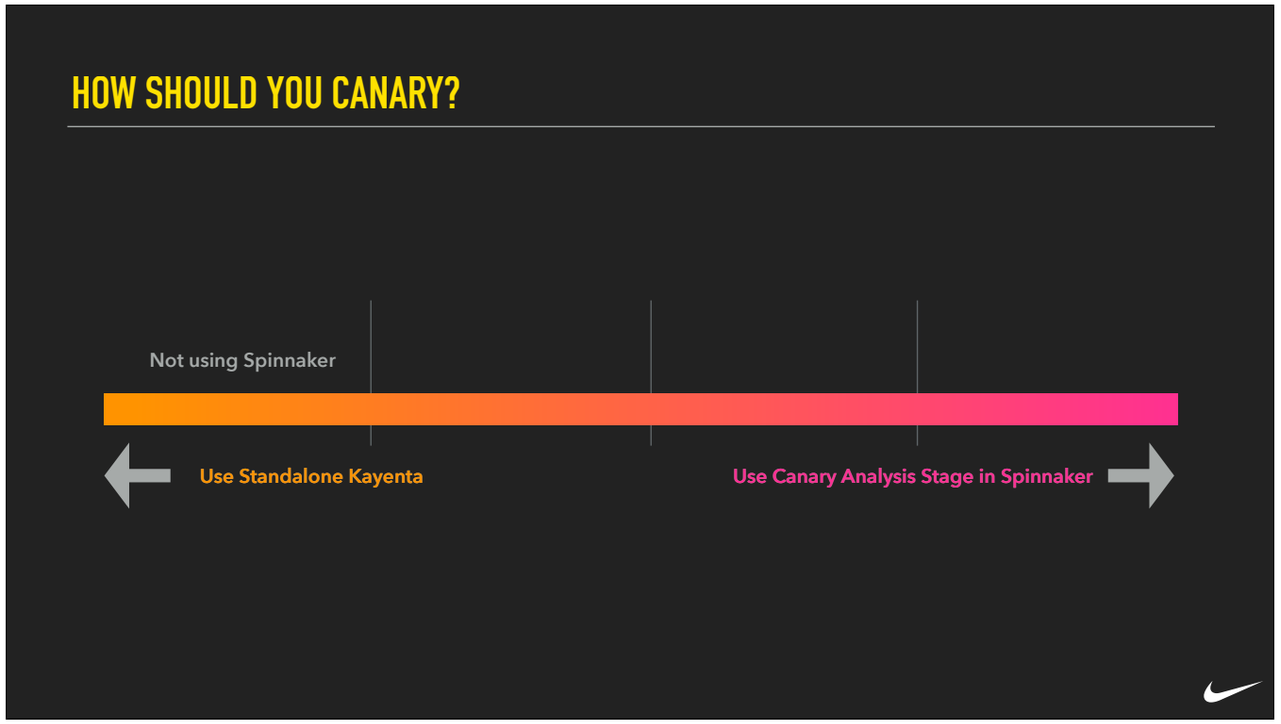
You can Canary now with Spinnaker, which has great first class support.

You can also canary by using Kayenta as a standalone service, which is the process we helped enable and will tell you about today.



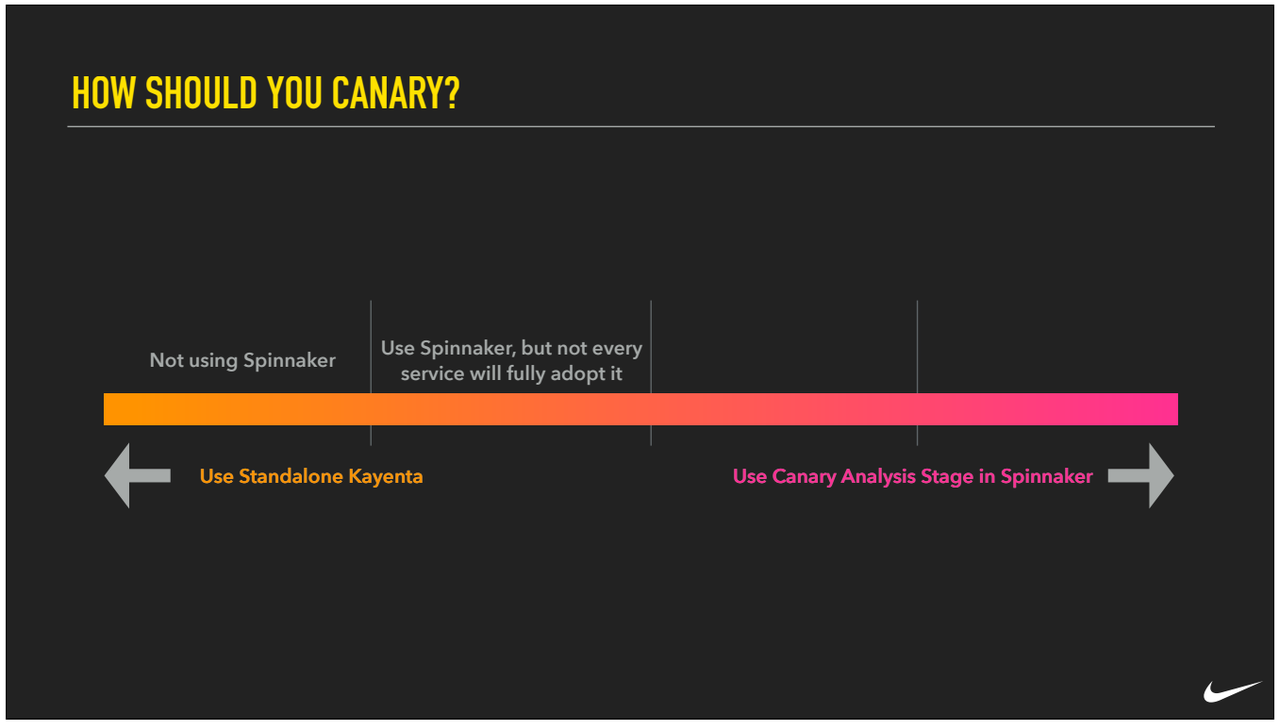
So which path should you take to canary, standalone Kayenta or using the canary features in Spinnaker?

There is a spectrum of scenarios where it makes sense to use Standalone Kayenta to where you would then be better off using the Canary Analysis Stage in Spinnaker.



The first scenario is when you have an organization that is not using or planning to use the complete Spinnaker solution.

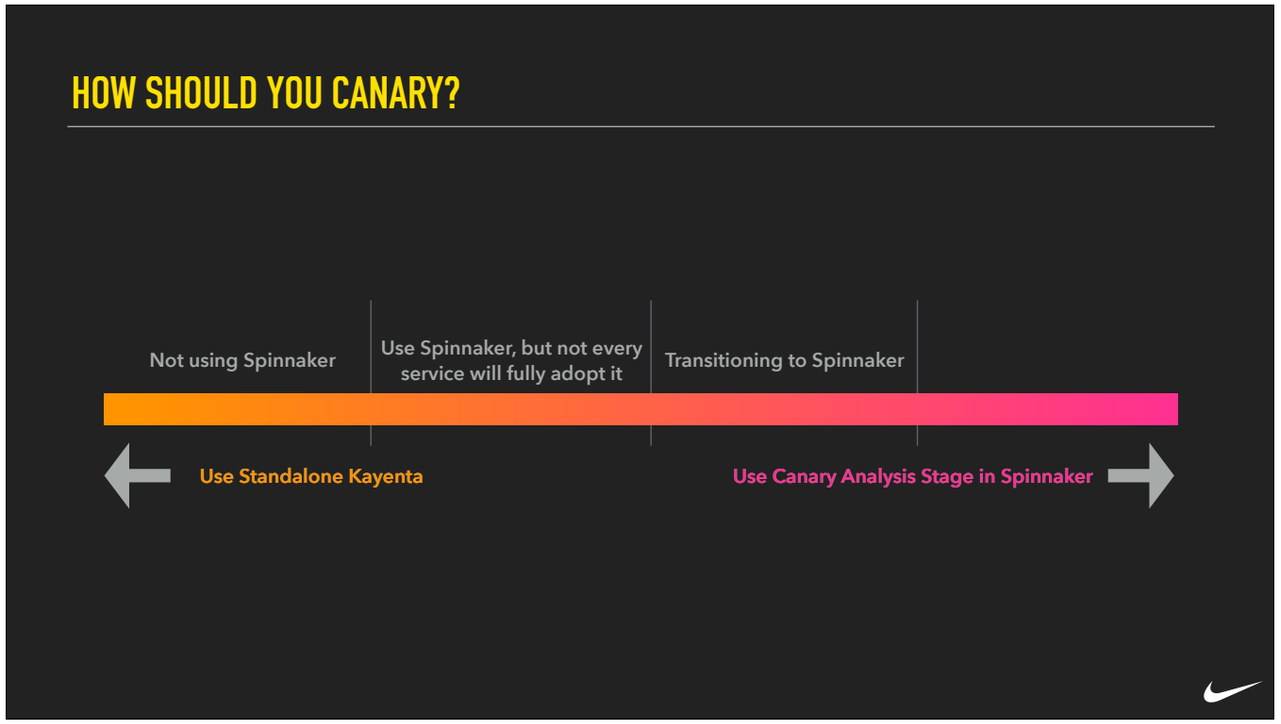
This group of people will get the most value out of Standalone Kayenta.



The second scenario is where you have adopted Spinnaker but are not planning on using it for all your frameworks / infrastructure.

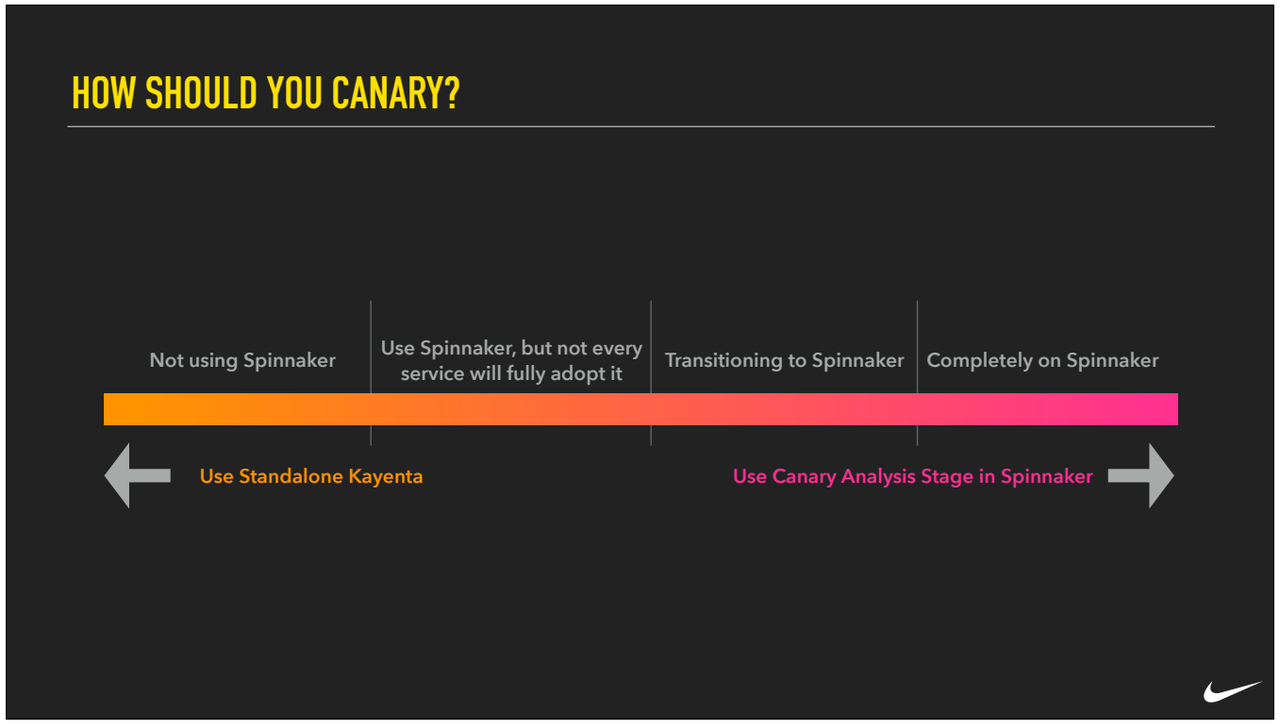
For example: You will use it for virtual machines but not for serverless or containers

This group of people will still get a lot of value out of standalone Kayenta.



The third scenario is where you are transitioning to Spinnaker but still have a bunch of legacy deployment processes around.

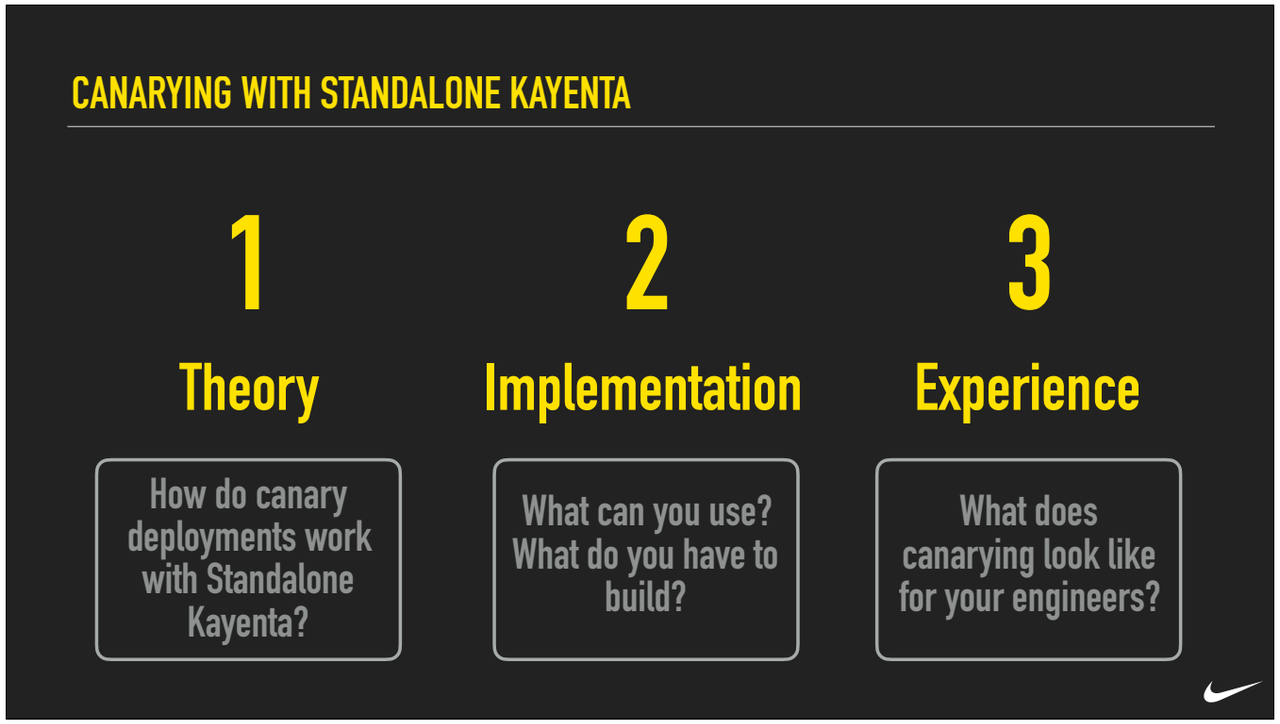
Depending on transition timelines it could make sense to use standalone Kayenta but you might be better off using the canary features in Spinnaker.



The final scenario is where you have gone all in on Spinnaker.

In this situation it doesn’t really make sense to use standalone Kayenta.

You will be better off using the canary features in Spinnaker.

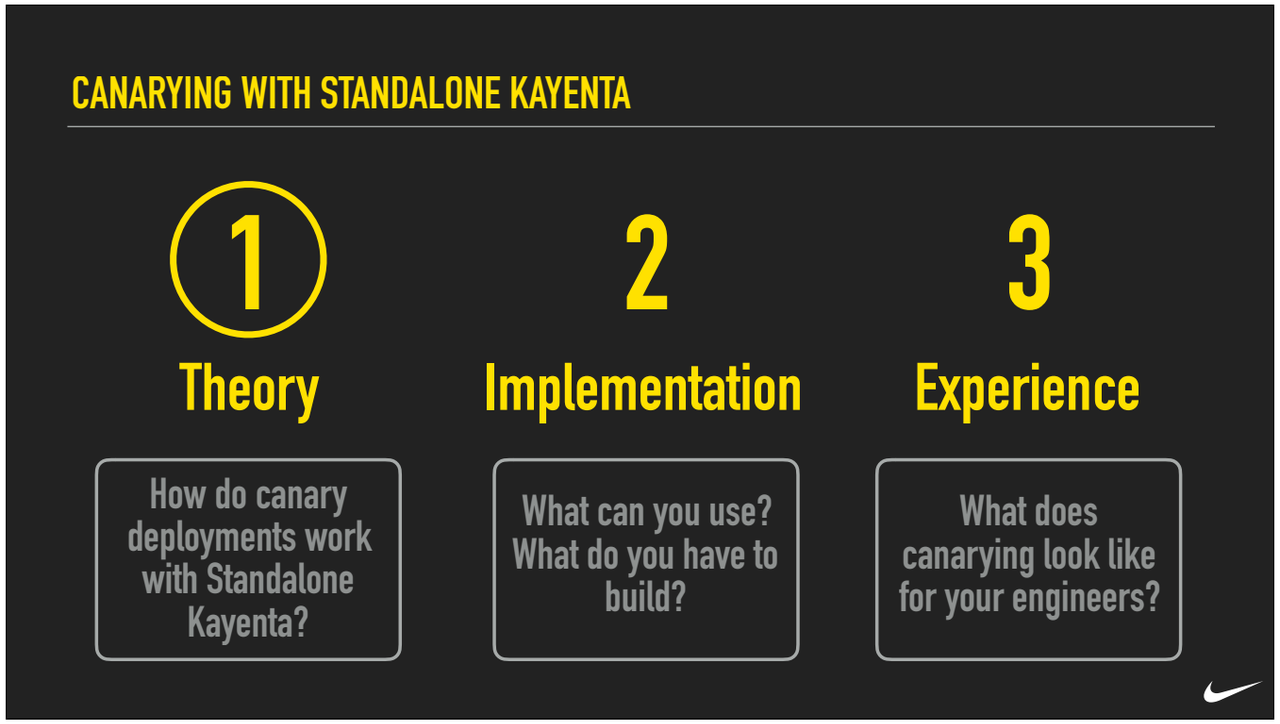


Lets drill into Canarying with Standalone Kayenta
We're going to talk about 3 things.

Theory:
How do canary deployments work with Kayenta

Implementation:
How does one implement Canary Analysis with Standalone Kayenta for an enterprise.

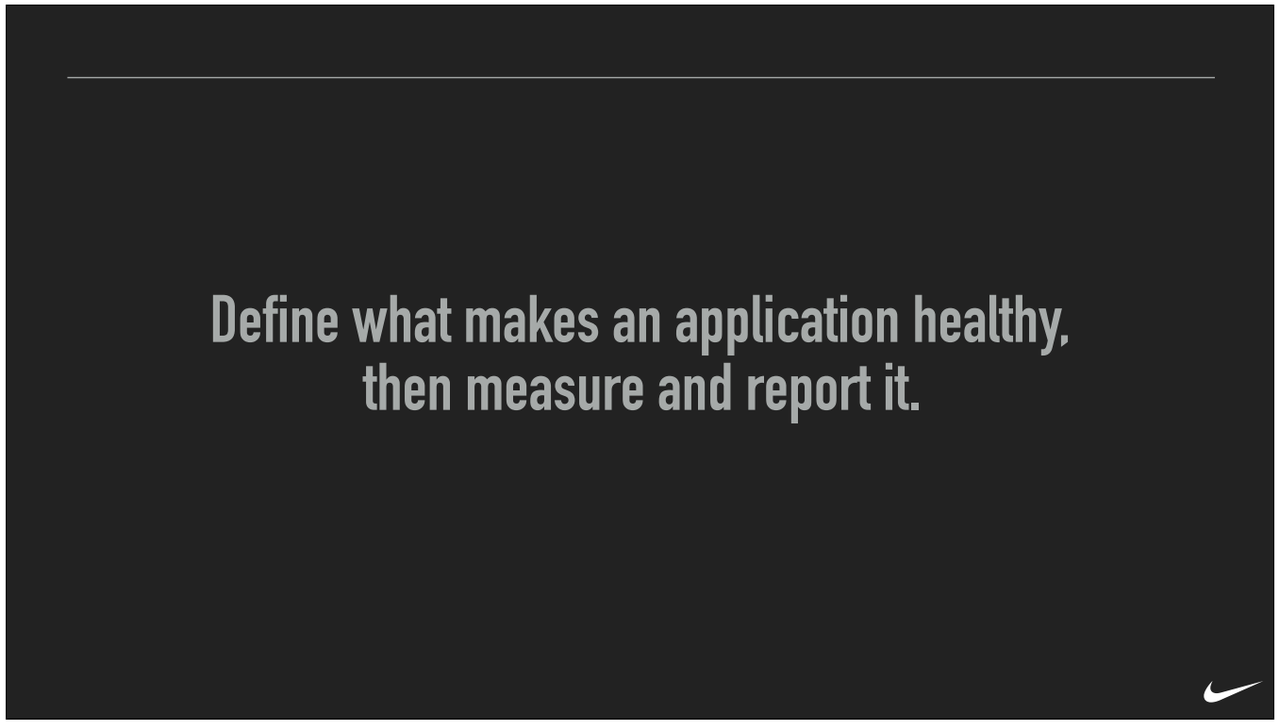
Experience:
When theory and implementation is combined, what does this unlock for the Users?



Let’s dive in..

Kayenta is a platform for canary analysis and ultimately judges whether a canary is good or bad.

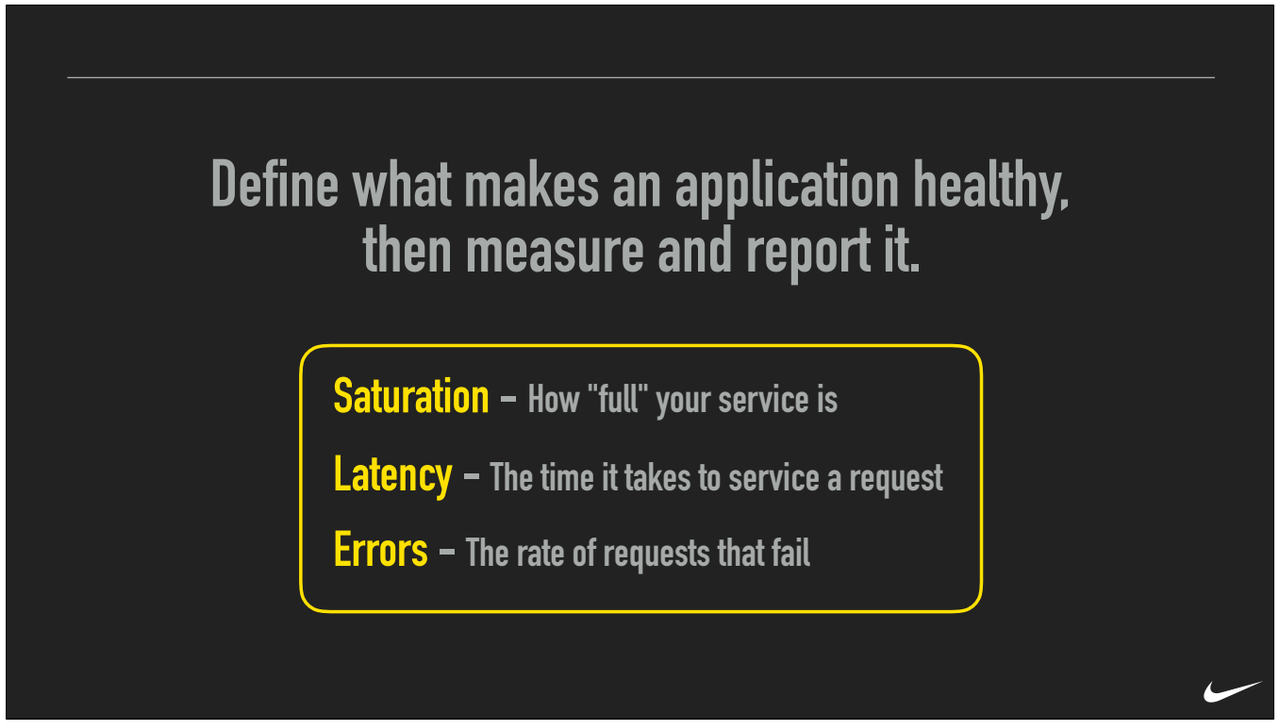
Let’s see how a canary deployment works with Kayenta



Before we can even start to think about canary deployments, we are going to need to be able to define what makes your application healthy, and then measure and report it.

It is important to define what metrics matter for your service, as this is the foundation for the Canary configuration that Kayenta uses to judge the health of canaries.

When creating canary configuration, the Spinnaker canary guide recommends that you use 3 out of the 4 golden signals of monitoring.



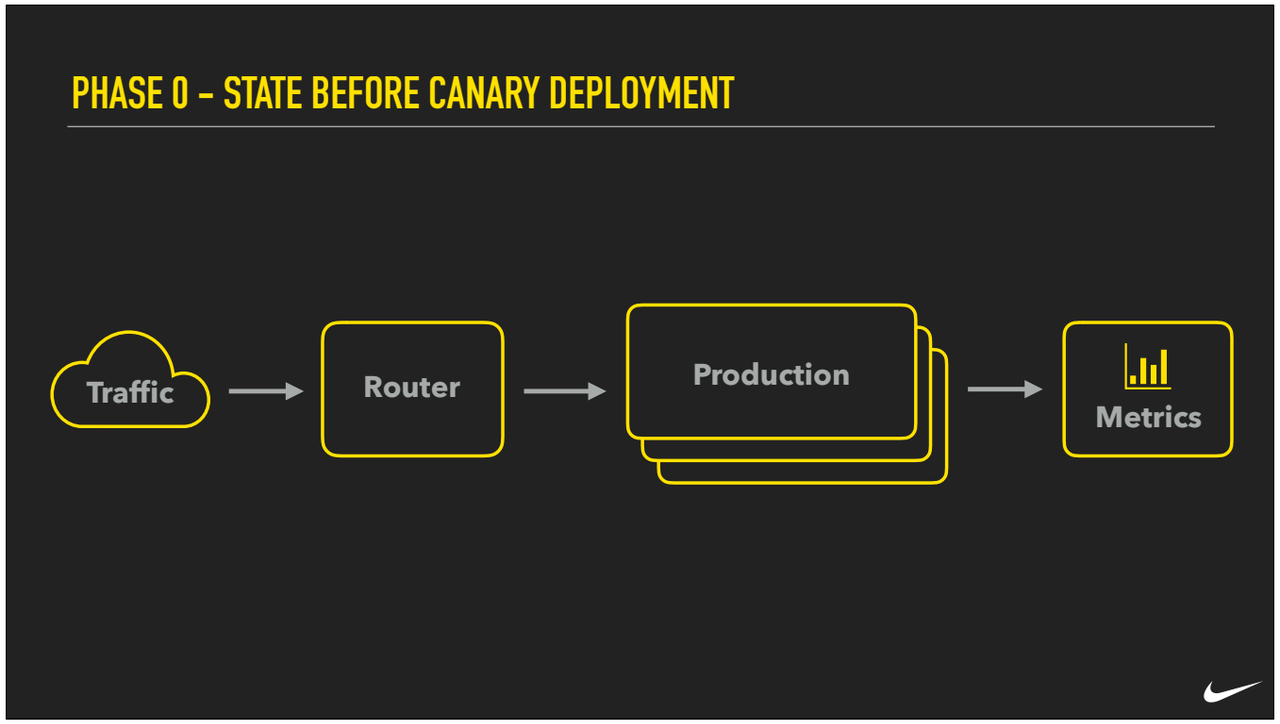
Those signals are:

Saturation
These metrics inform how full your service is.

They are metrics such as CPU, memory, disk and network usage.

Latency
These metrics inform how long your service is taking to fulfill requests

Errors
These metrics inform the rate at which requests are failing

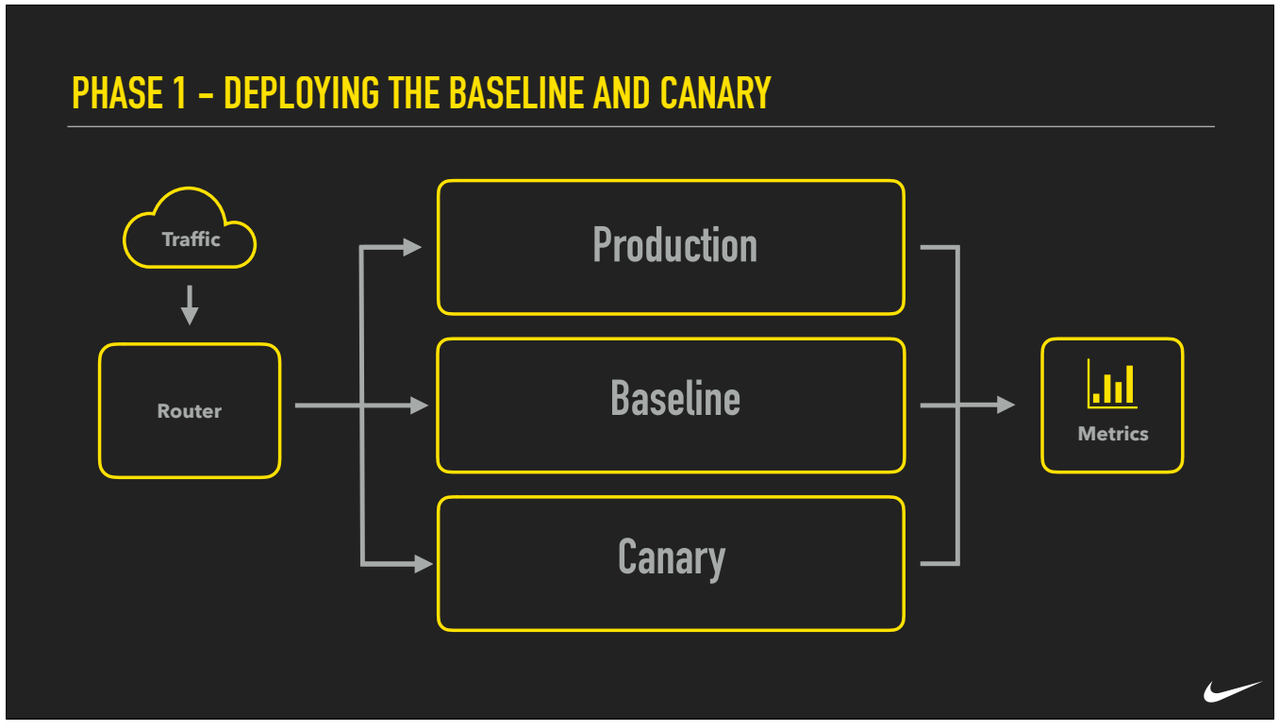


Okay now we can start thinking about canary deployments.

Lets take a look at the state of our application before a canary deployment

We have traffic being routed to our production server group

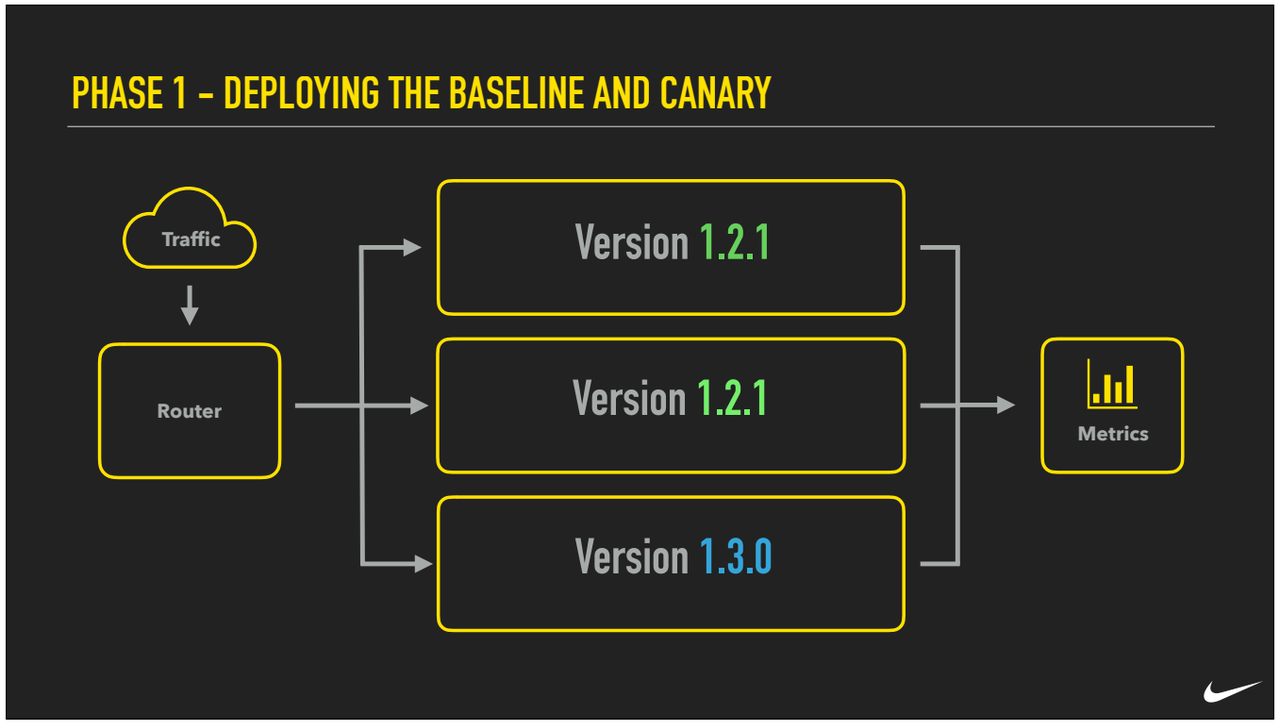
All the instances in that group are reporting their metrics to some metrics stores such as: Signalfx, New Relic, or Prometheus.



Phase 1 is where we deploy the baseline and canary groups.

This is where we let a percentage of traffic flow to our experiment and see if our little canary lives or dies.

But what does this mean?
What is the baseline and canary?



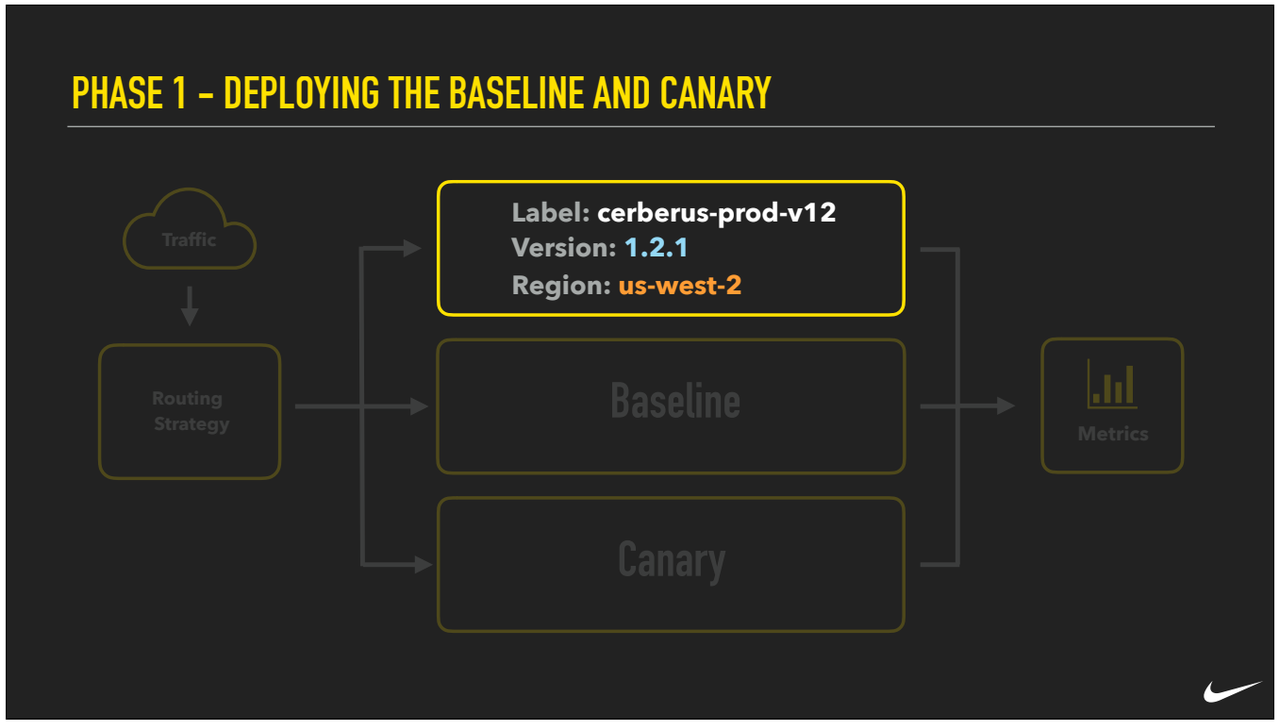
The baseline is a fresh version of what is current deployed in production.

This can be a baked AMI or a Docker image, some sort of deployable artifact.

We do this to eliminate cold boot issues such as cache warming and help make metrics between the baseline and canary equivalent.

Whereas the Canary is the new deployable artifact

So as you can see our production and baseline groups are both at version 1.2.1 but the canary is running the new version of our application 1.3.0.

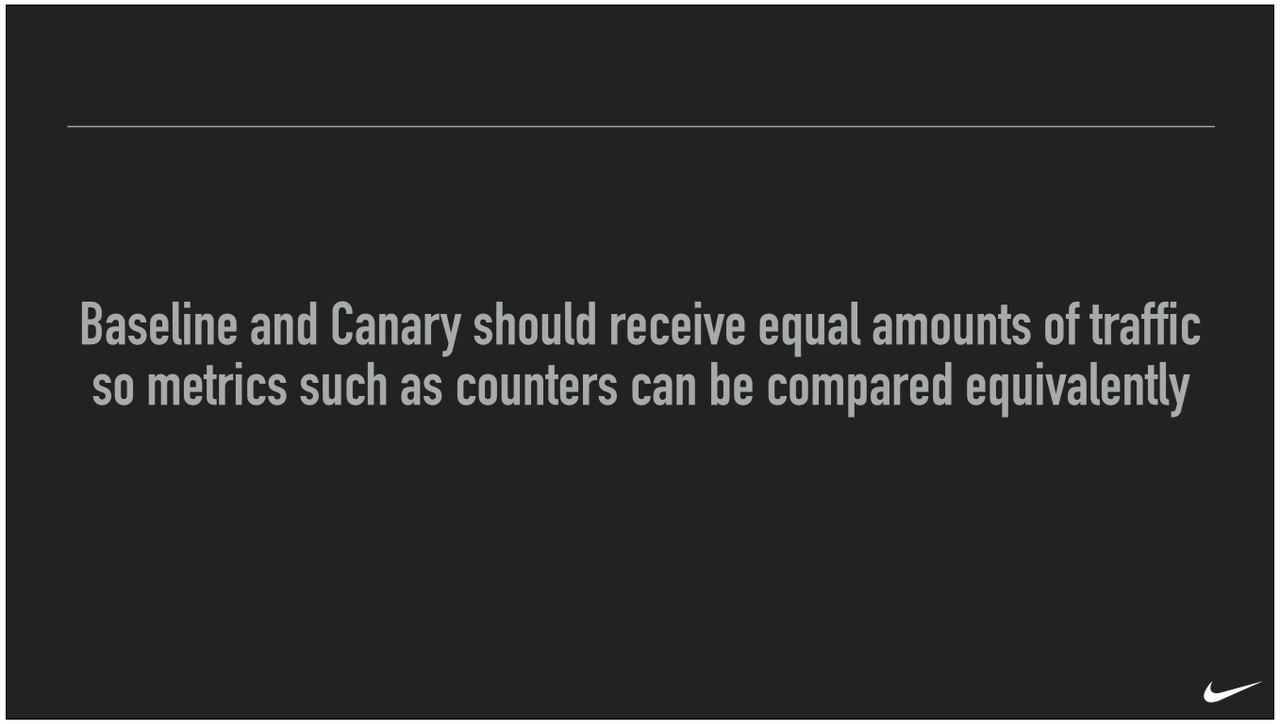


In this situation we have multiple server groups deployed with the same application version running.

We can’t just use the version to differentiate the existing production group from the baseline.

You need to make sure that you are sending metadata with your metrics that includes a unique id for each server group to your metric source.

This is so that metrics for the baseline can be queried separately from the current production server group.

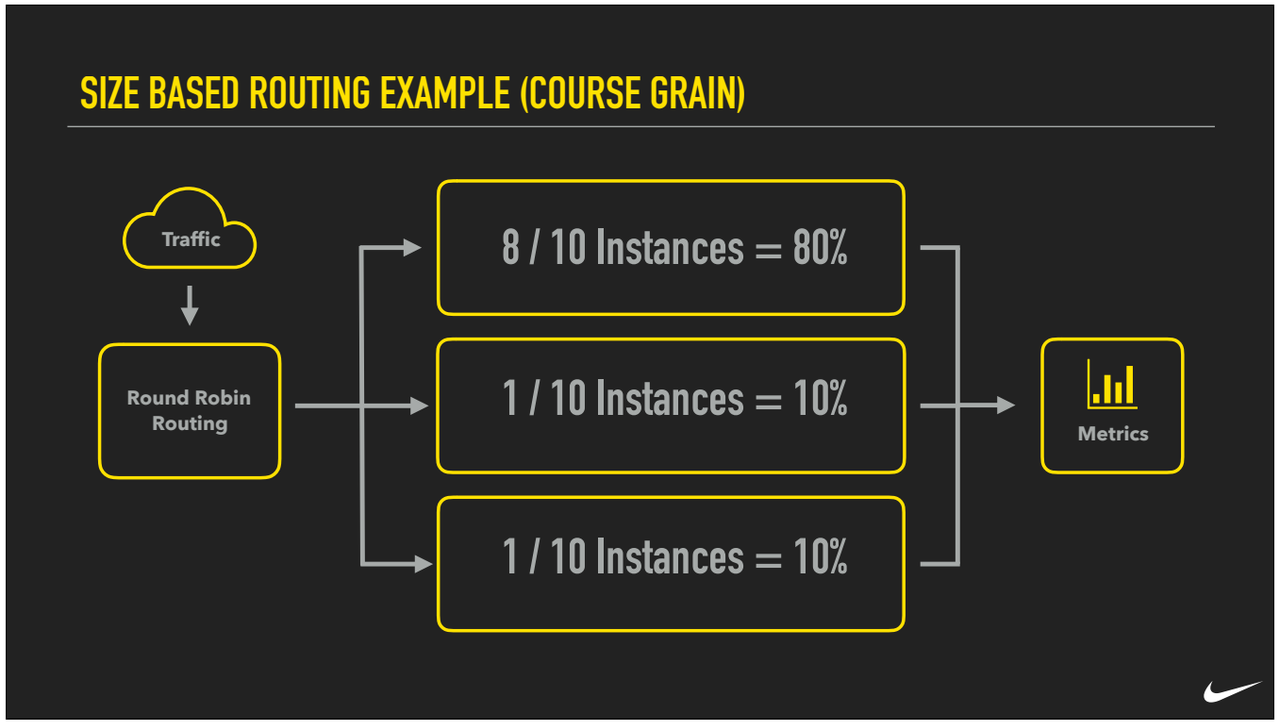


The baseline and canary need to receive the same amount of traffic for certain types of metrics to be equivalent.

Metrics such as error counts would be meaningless in scenarios where the volume of requests are different.

You could maybe convert these metrics to rates, but then saturation metrics such as CPU% and memory usage start to get weird.

Let’s take a look at 2 routing strategies



The first strategy is: Size Based Weighted Routing

We sometimes refer to this as coarse grained routing since the weights are controlled by instance counts.

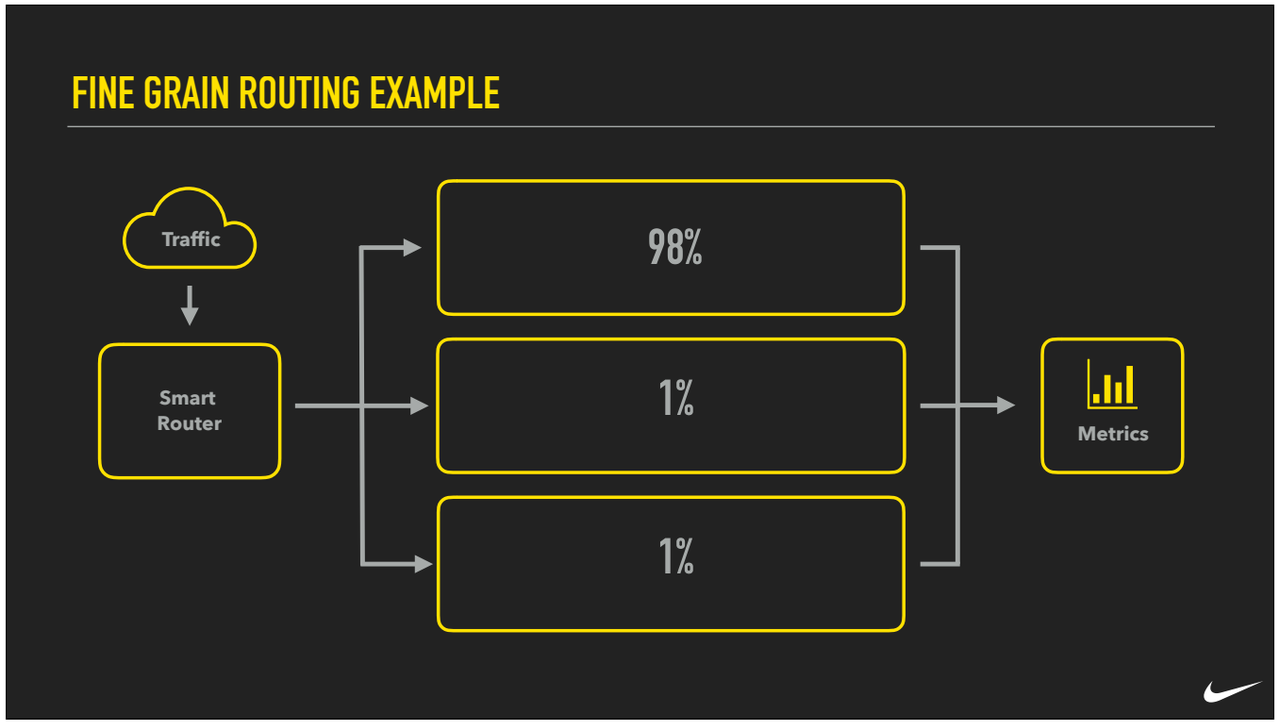
This strategy is the most common one we have come across because it's the simplest.

The strategy is to round robin requests through a load balancer or on the client in the case of service discovery.

Explain the diagram:

As you can see in our diagram 80% of the incoming traffic is going to the already established production server group.

Since we have 1 instance for the baseline and 1 instance for the canary, 10% of the traffic is flowing to each.

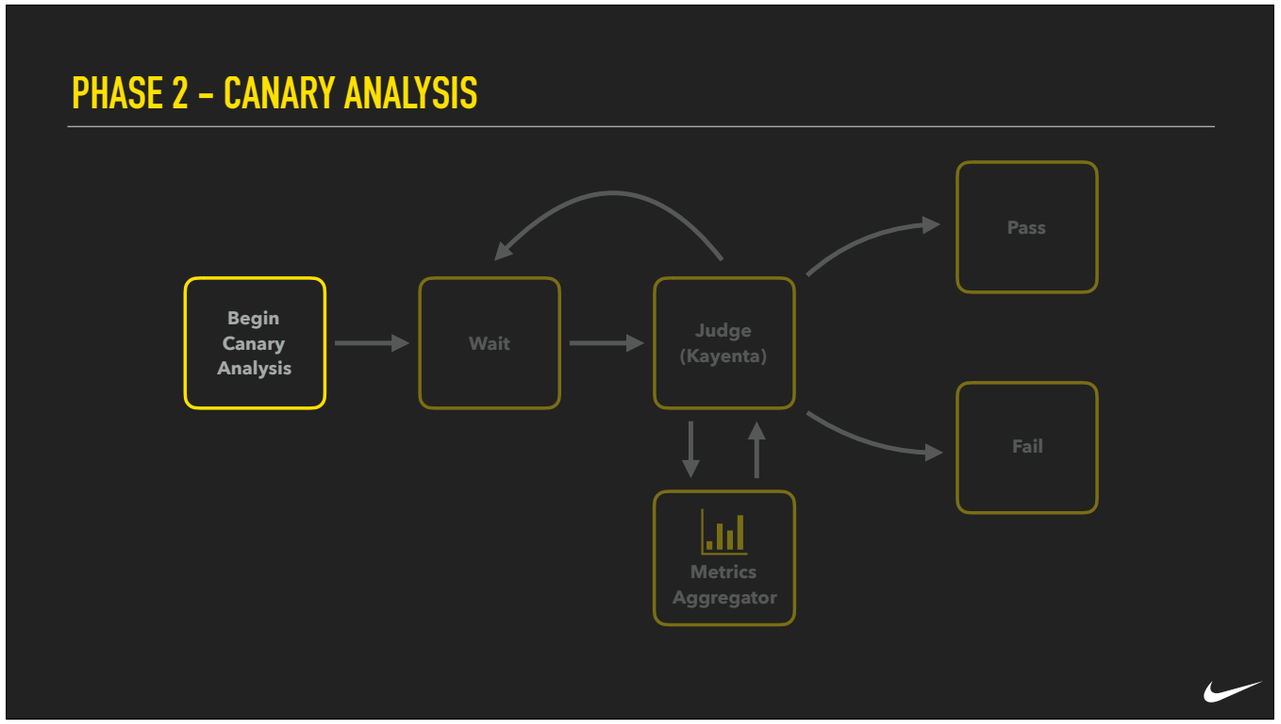


The second option is: Fine Grain Weighted Routing

This strategy relies on something smarter than round robining and allows for the setting of specific percentages of traffic.

AWS API gateway for example can do this.

However, this potentially involves a more complicated deployment process.



Now, phase 2: we have our infrastructure set up.

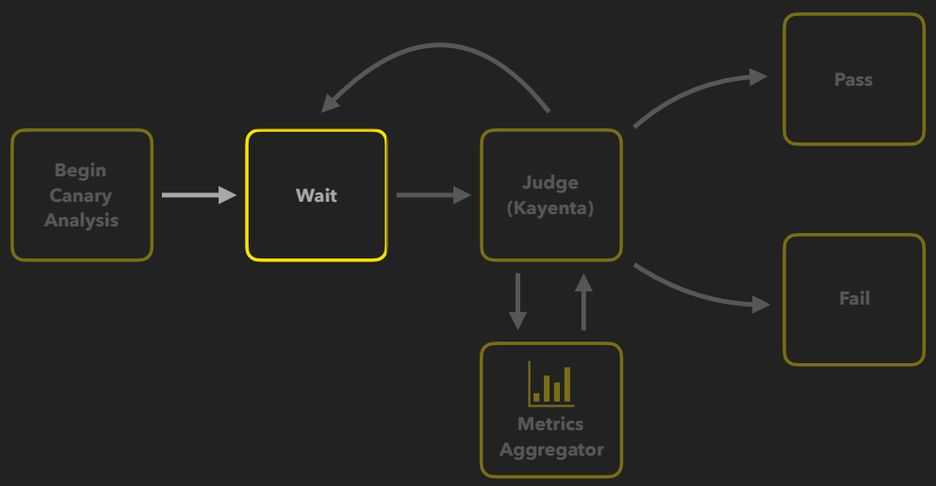
We have deployed the baseline and canary.

The baseline and canary are taking equal amounts of traffic.

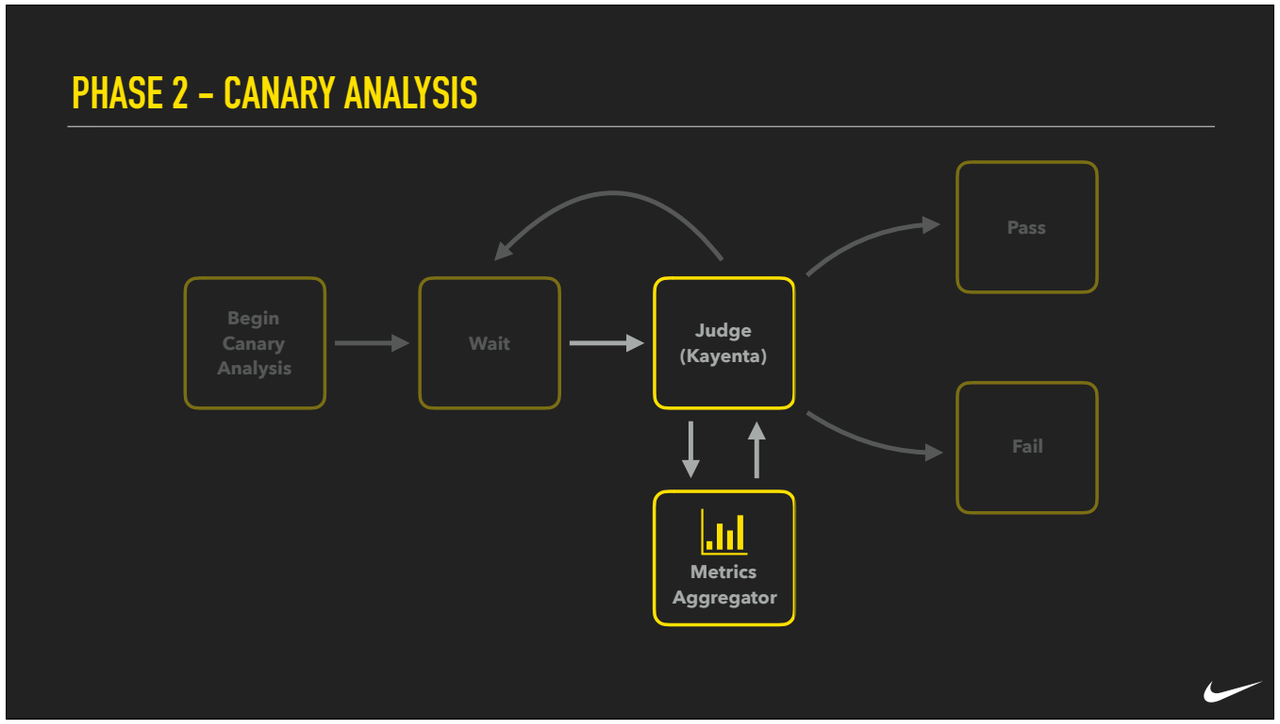
The baseline and canary are reporting metrics with metadata that allow for their metrics to be queried independently,

We can now do the actual canary analysis phase of the deployment.

PHASE 2 – CANARY ANALYSIS

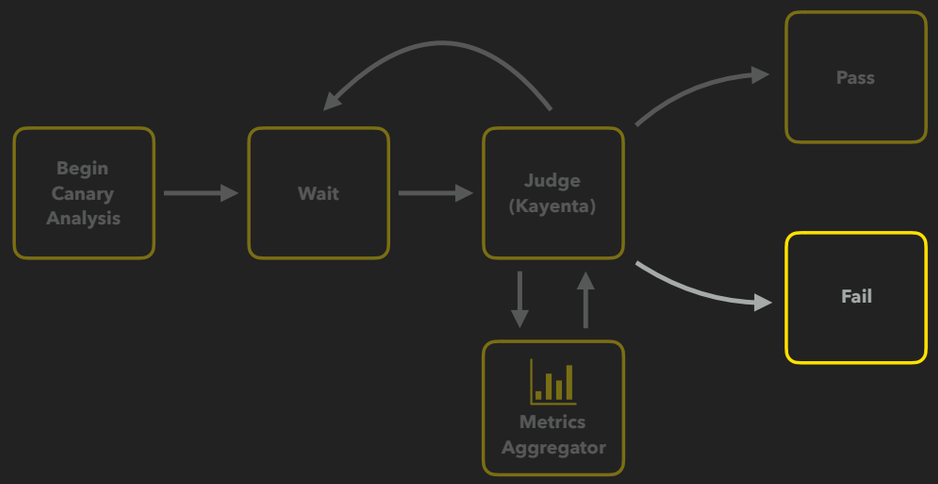


The first step of this phase is to wait and allow traffic and metrics to flow through the system.

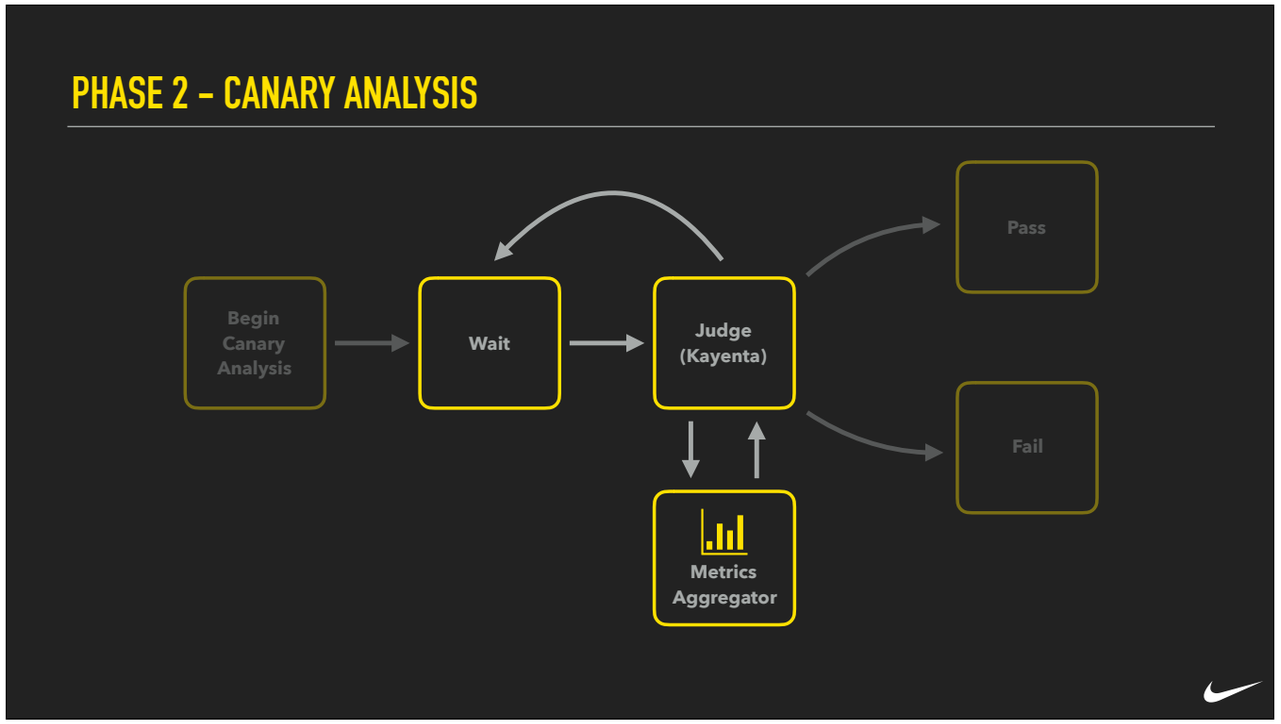


After some amount of time we can query the metrics source for the baseline and canary's metrics and do a statistical comparison to see if the metrics for the canary have changed for the worse.

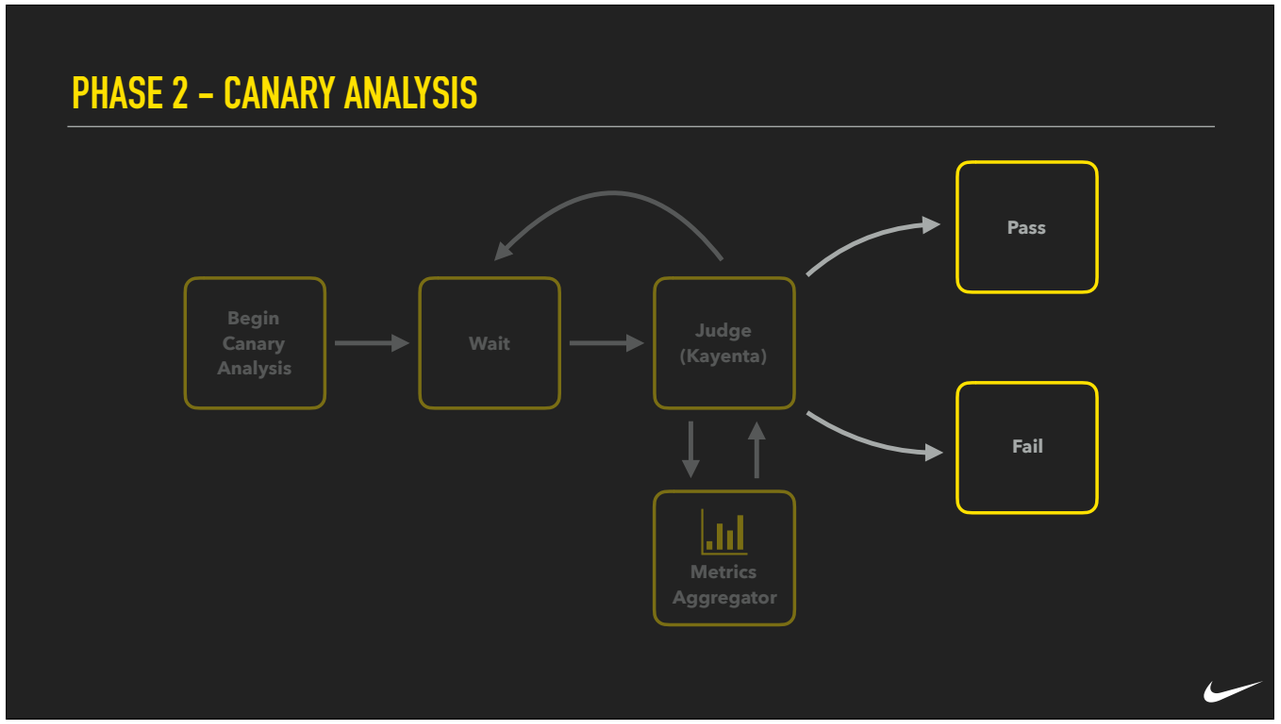
PHASE 2 – CANARY ANALYSIS



If the metrics for the canary are bad, we can fail fast and exit the canary analysis phase with a fail status



If the metrics for the canary are within the configurable thresholds we can continue with another wait and judge cycle and repeat this process for n intervals over some lifetime.



At the end of the process we make one last judgement and produce a pass or fail result.

This result can then be reacted to downstream whether that's automatically rolling forward with the new change or rolling back or even allowing for a manual user decision.

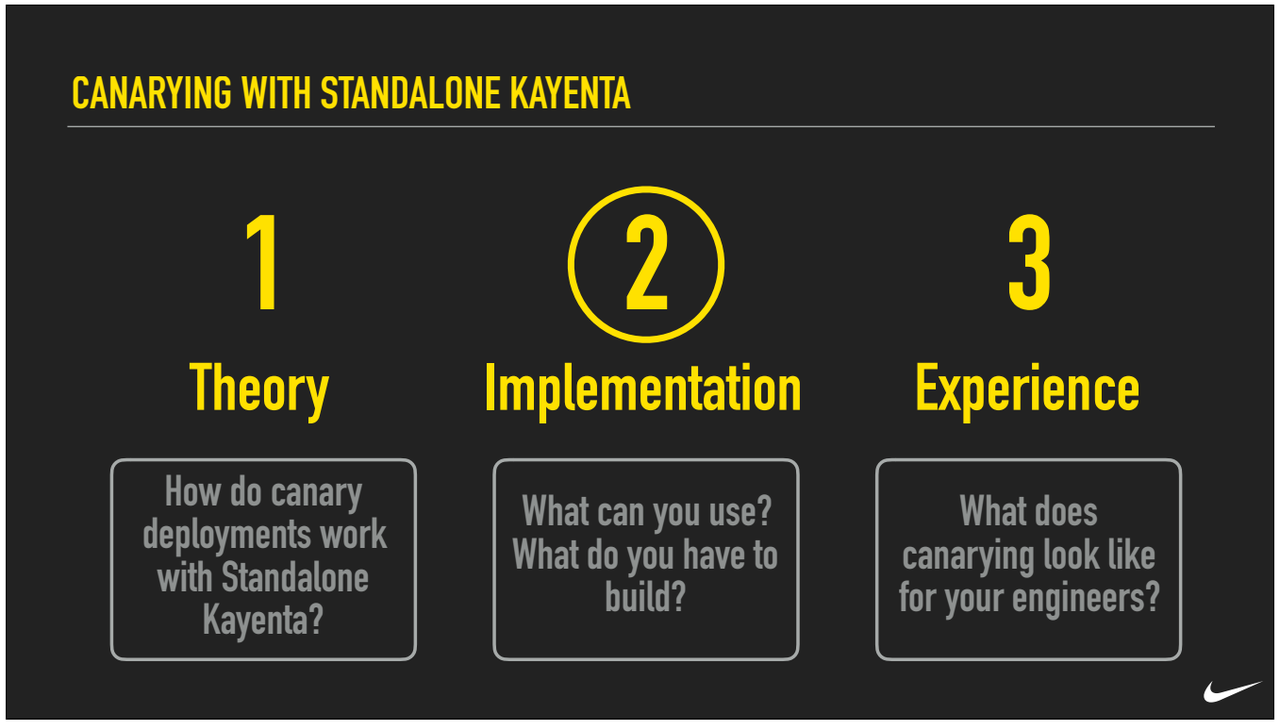
Let’s recap how a canary deployment works with Kayenta

We instrument our app to report the golden signals and use those later to determine the health of a canary.

We deploy the baseline and canary along side the existing production server group

We do canary analysis over some lifetime and perform multiple judgements, failing fast if need be.

Finally we get an aggregated judgement and react to the results



Justin: Melana, all of this theory is great and all, but how do you get it to work?

Melana: Well it is just a simple matter of programming. *joke*

Let’s talk about what you can use to canary with Standalone Kayenta, and then what you have to build.

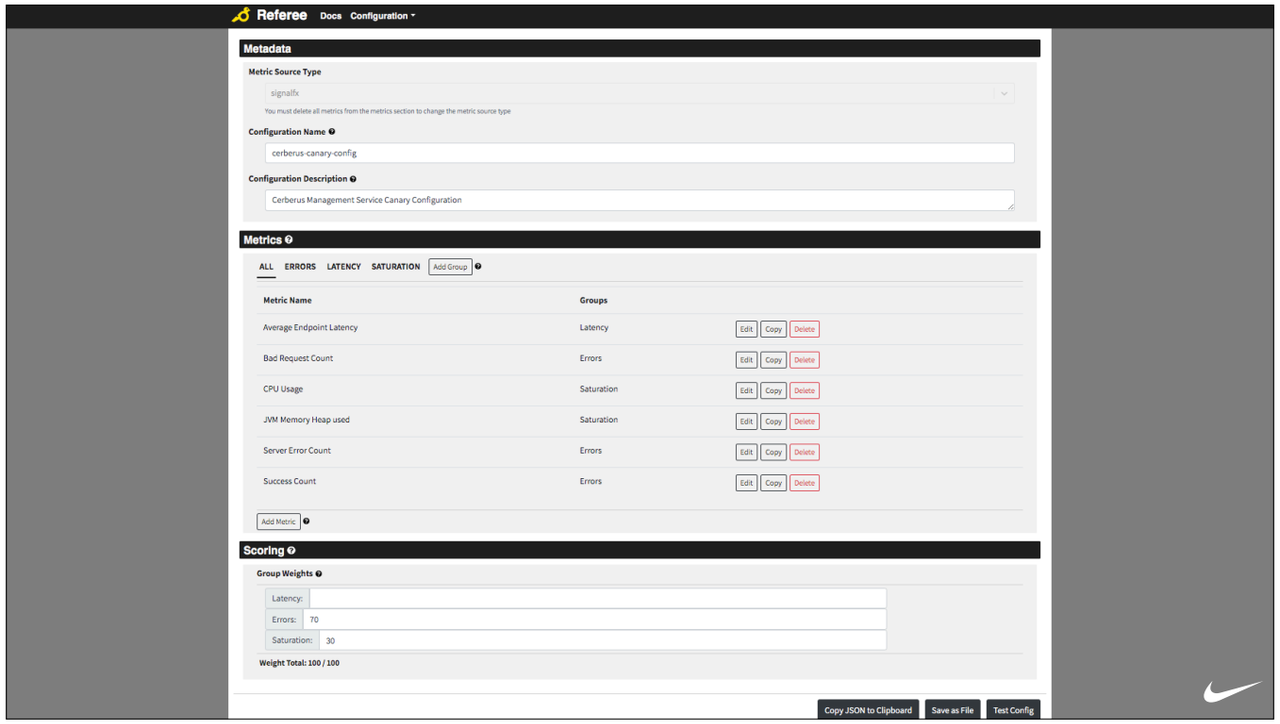
So what you can use in terms of open source software to canary?

We built Referee, an open source UI for viewing reports and using developer productivity tools.

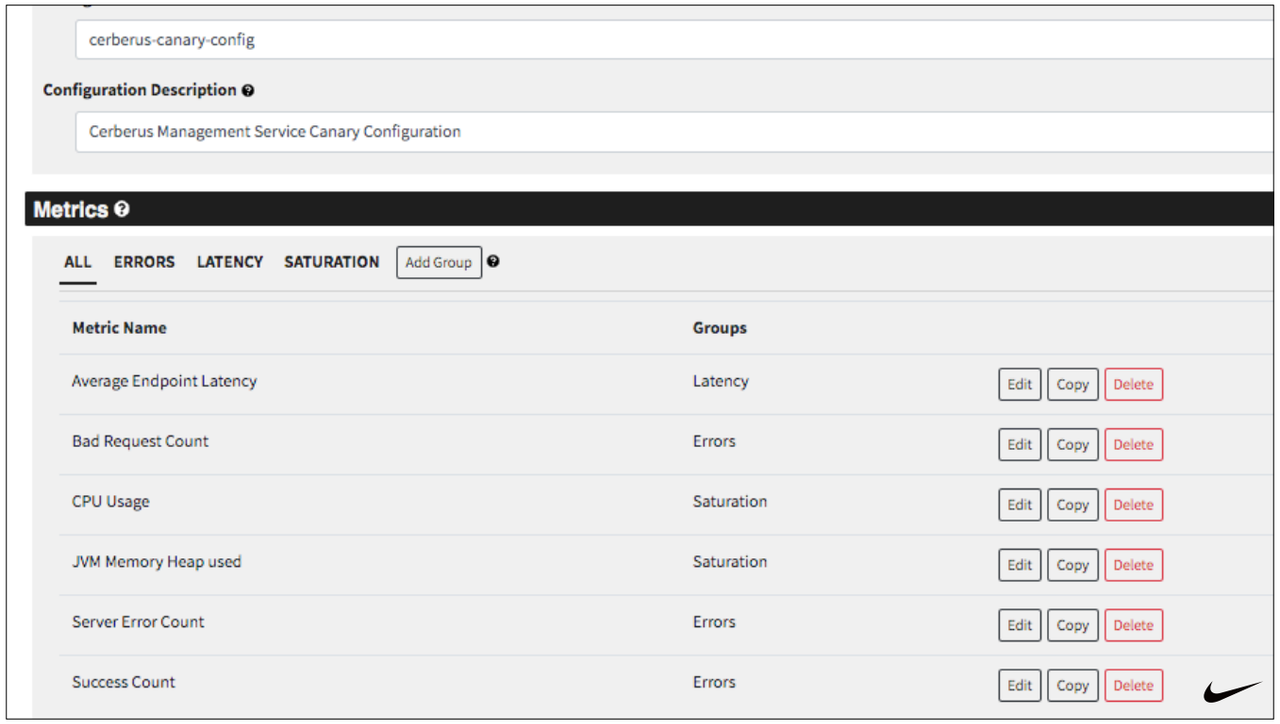
We had realized we needed an additional layer for our customers to be able to process the results from Kayenta.

We tried to get Deck, Spinnaker's UI, working for just Kayenta but it is too tightly coupled with the complete Spinnaker solution, so we built our own UI that's just for Kayenta.

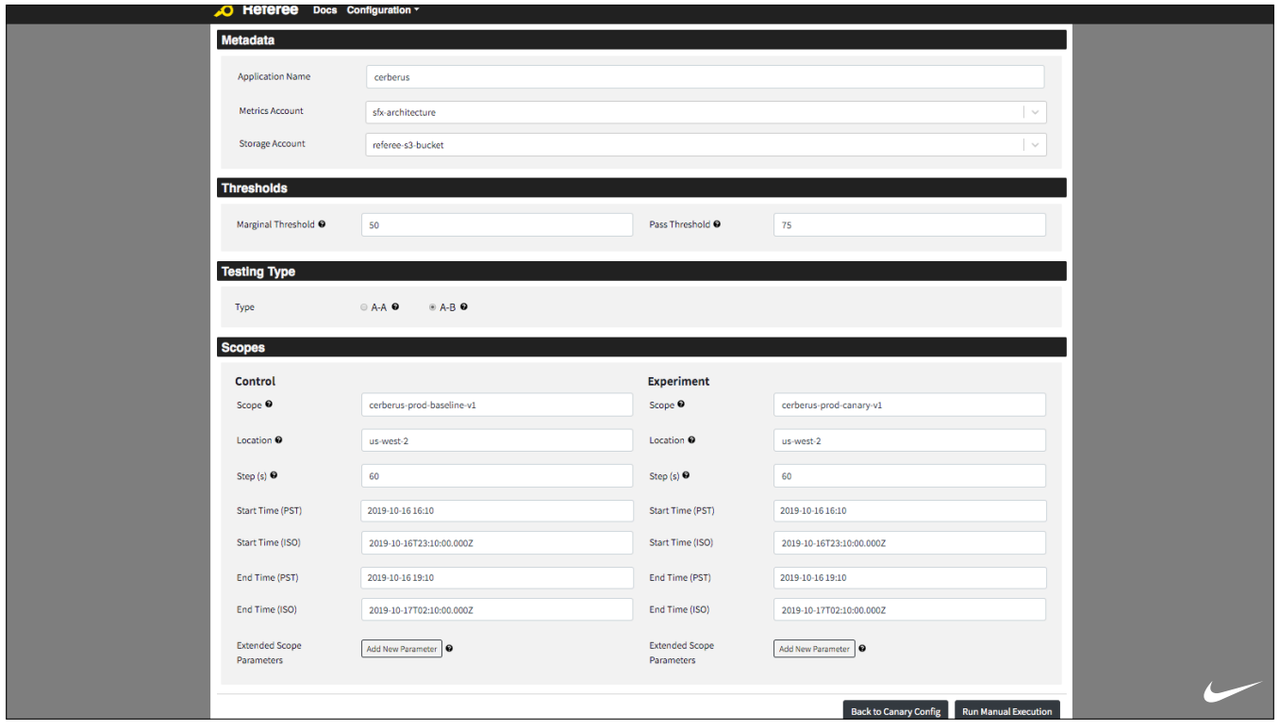
Throughout the whole process, we partnered with members of the Spinnaker UI community and engaged and collaborated on designs and implementation ideas with them. We couldn't have done this work without their support.



Referee includes a canary configuration generator tool.
To enable teams to build their canary config
A lot of this is based off of the user experience in the Spinnaker Deck UI

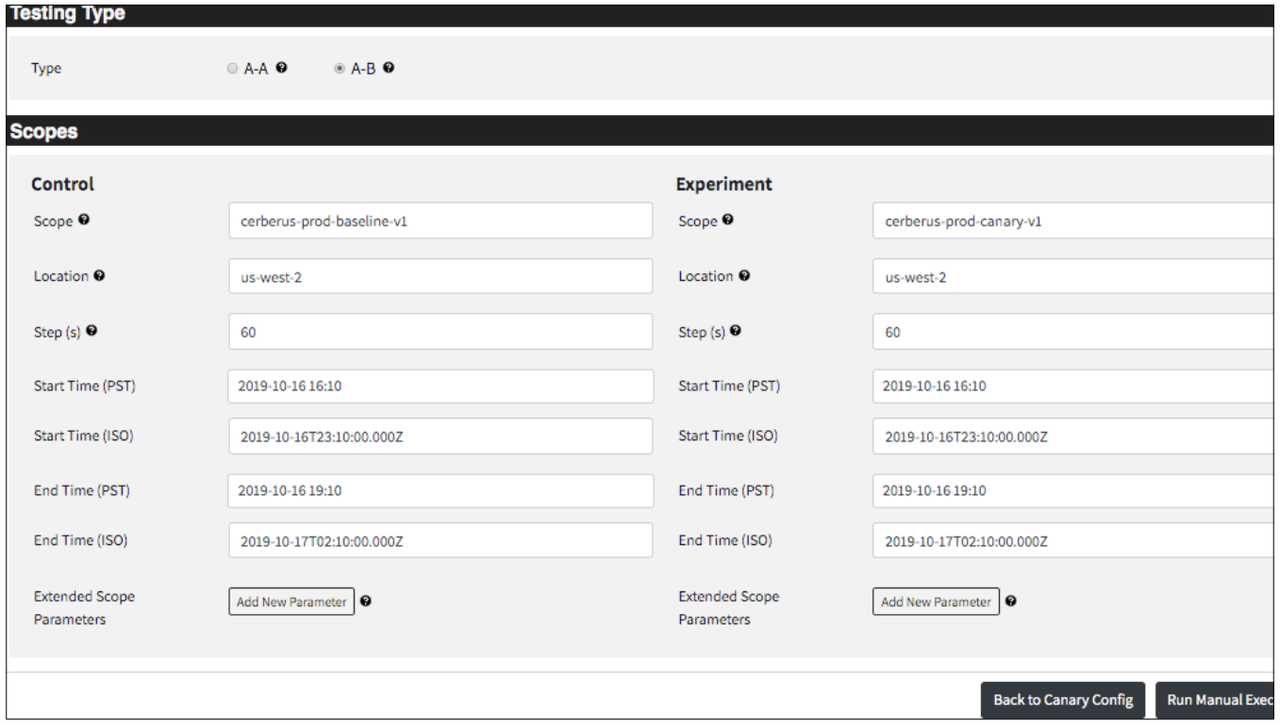


As you can see in this canary config, some of the golden signals of latency and error counts are used as metrics to indicate the health of the service.

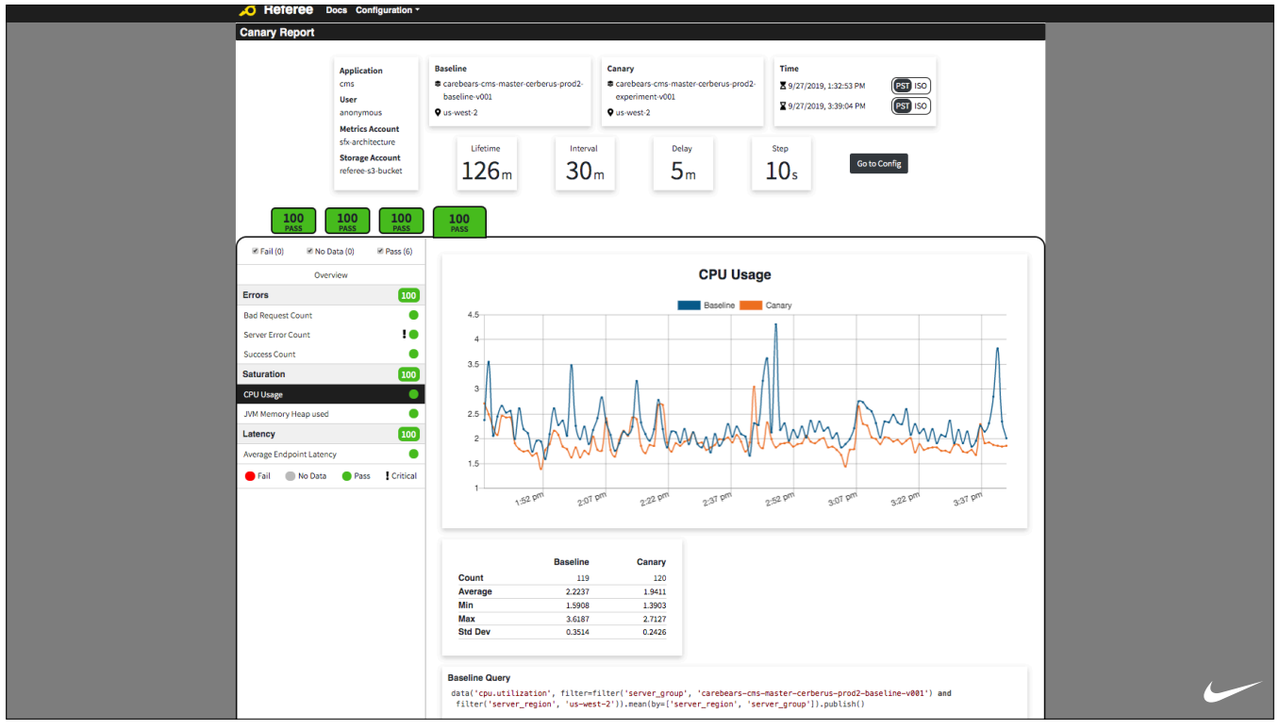


Then there is a retrospective analysis tool.
We built this tool to enable teams to rapidly iterate on their canary config.

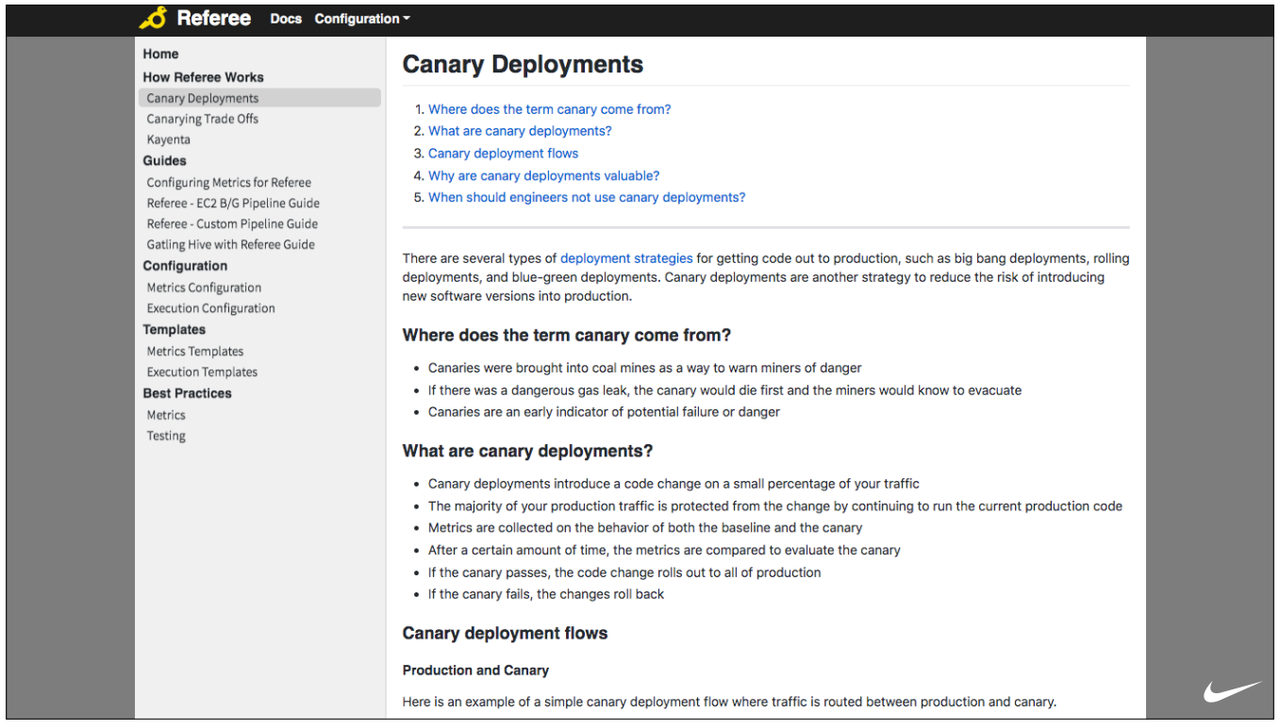
Retrospective analysis is a way to conduct canary analysis on historical events or data. It can be used to speed up testing, so when users test small changes in their configs, they don't have to wait for a full canary run every time.



In this example, we can see the scope is two different server groups, one, the baseline and one, the canary.



Then we built a report viewer for users to see their canary results.
In this example, we can tell from the green along the left hand side that the canary passed and all the metrics look healthy.



Finally, we also have an internal documentation site in Referee, so you can host docs specific to your organization in the same place as the other existing tools.

In addition to this custom Kayenta user interface, we made some contributions to Kayenta itself that Justin will walk you through.

OPEN SOURCE CONTRIBUTIONS

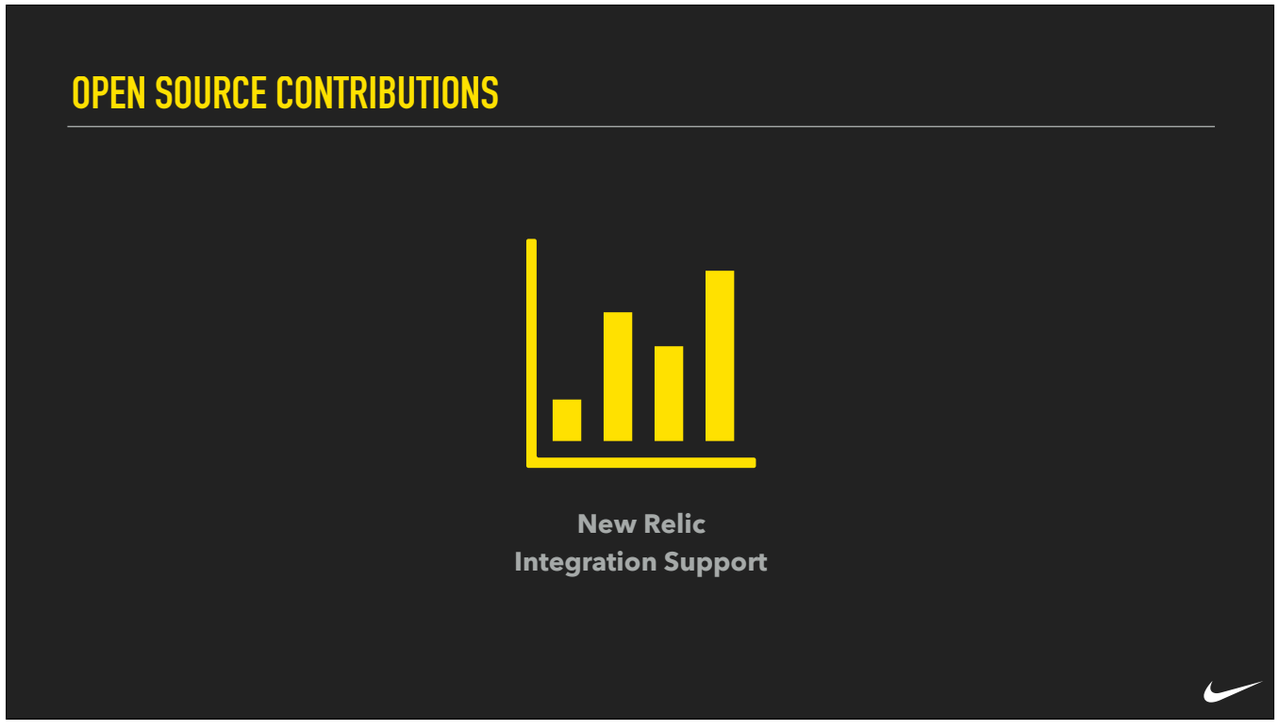
SignalFx Integration

Our engineering teams use SignalFx to monitor and report those golden signals.

Kayenta didn’t have SignalFx support so we added it.

We also added support for SignalFx in Deck, Spinnakers UI, so that the OSS community can use SignalFx as well.

Lastly we added SignalFx support to Halyard, the CLI for managing spinnaker, so that the community can configure signalfx when managing their spinnaker instances.



Our engineering teams also use New Relic and it was on our list of metrics sources to contribute upstream.

So when we saw the community open a PR for the initial integration of New Relic, we helped review that and contribute additional features to the user experience in Kayenta.

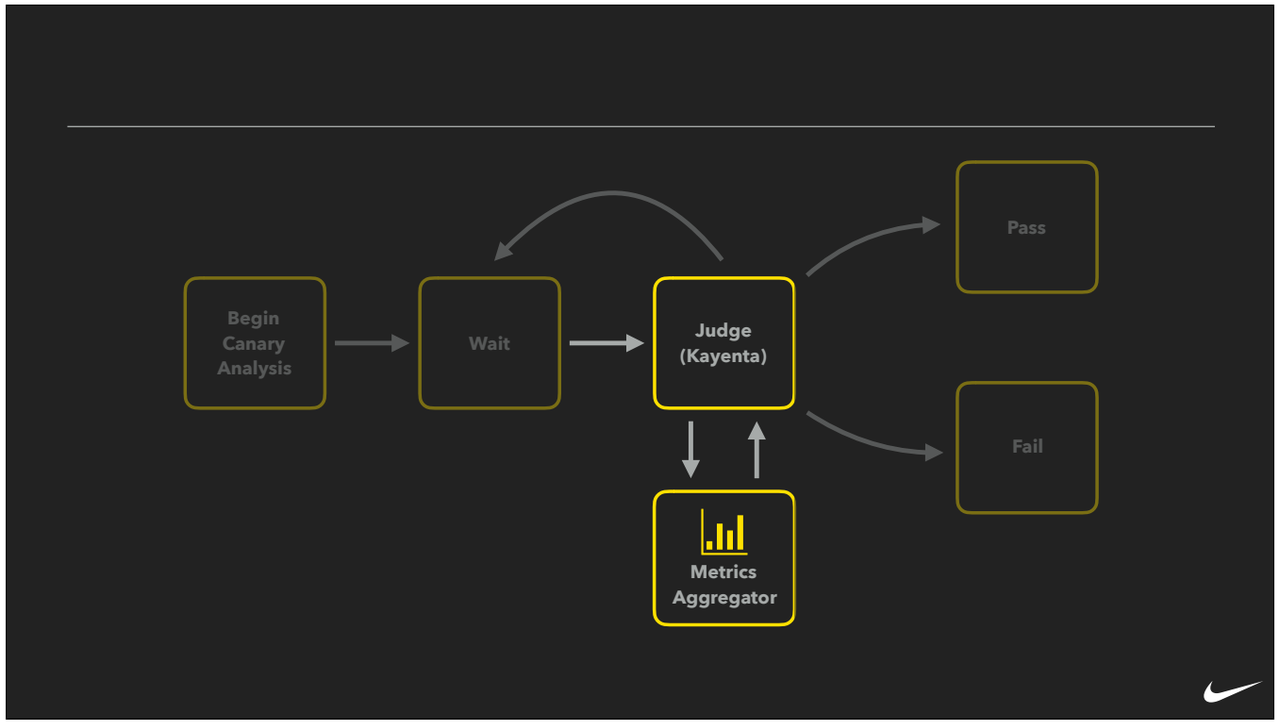
OPEN SOURCE CONTRIBUTIONS

Kayenta Standalone endpoint in API

We we did our initial Canary research with Spinnaker we liked the user experience that the canary stage offered.

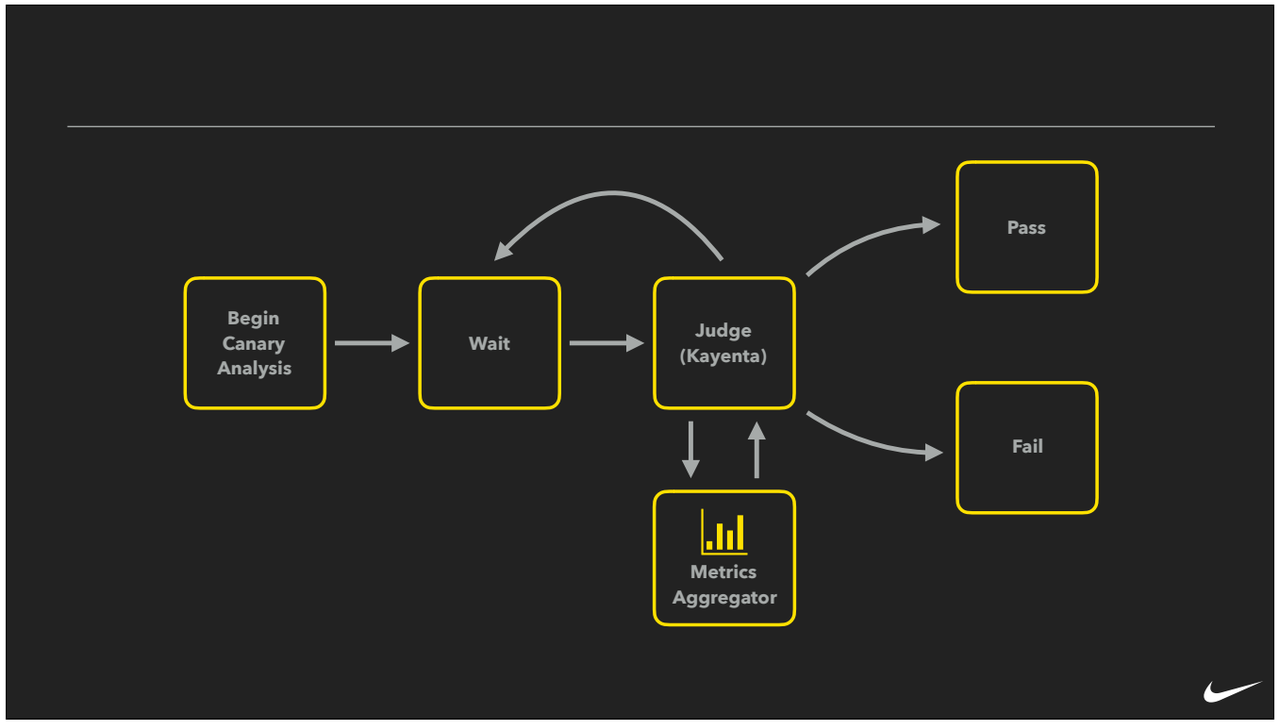
That stage is a user experience that is spread across multiple microservices.

We created a new endpoint in Kayenta consolidated that user experience into a single user callable API endpoint



If you recall our phase 2 process chart for how a canary deployment works with Kayenta.

Kayenta originally just had an endpoint that enabled the fetch and judge portion of phase 2.



The rest of the phase 2 logic was in another microservice called Orca, that you triggered via a Spinnaker pipeline

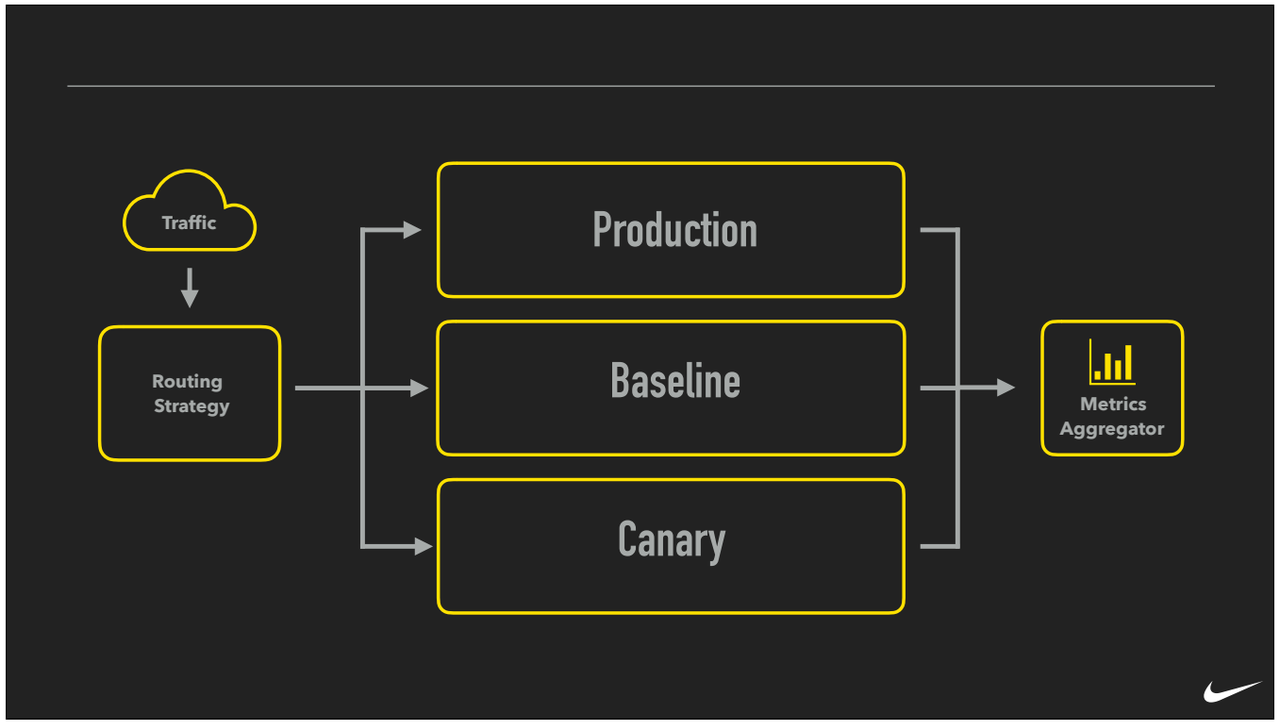
But with the new standalone endpoint you can do that aggregated analysis over multiple iterations using just the API in Kayenta.

We had this originally implemented as an internal Kayenta plugin but we noticed lots of people in Spinnaker Slack and Github asking about this so we contributed it upstream.

That covers our contributions, Melana is going to walk you through what you have to build to integrate all this in practice.

Using Kayenta is not as simple as making an API call.

If you are using Spinnaker, there is a lot of integration work built in for you. However, if you are using Standalone Kayenta, there are some technical challenges you will have to figure out at the enterprise level.



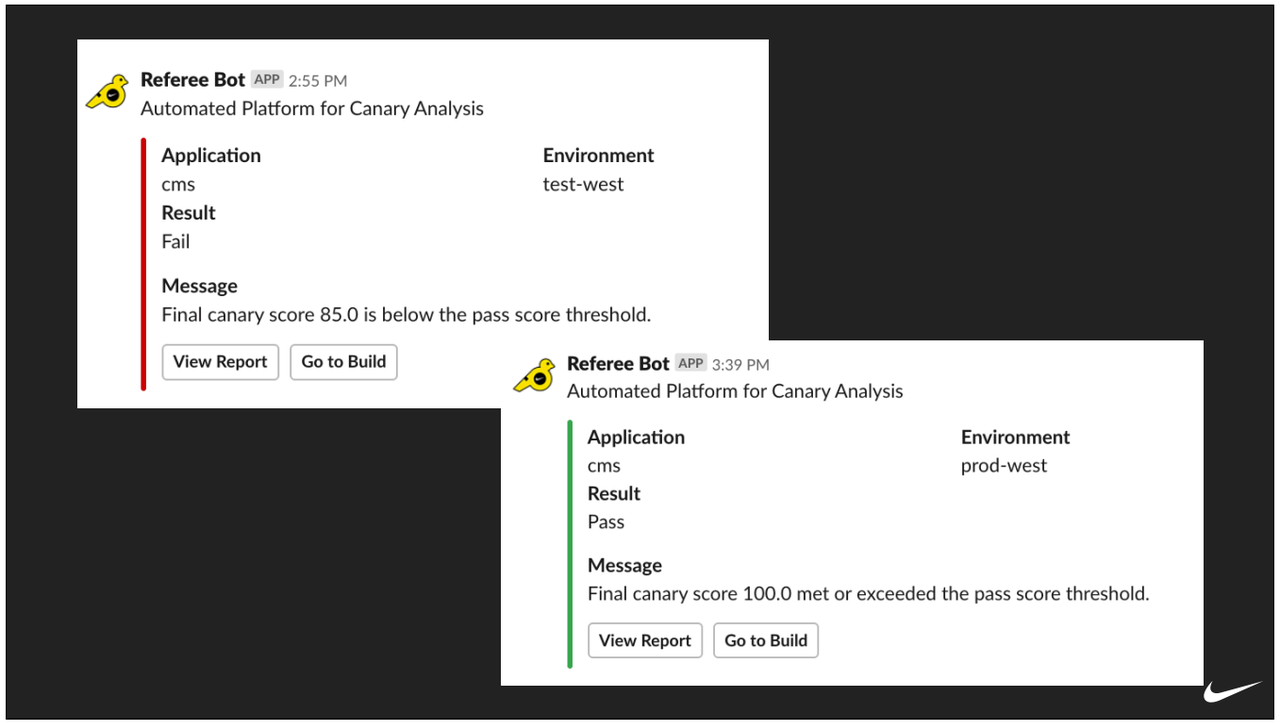
To use standalone Kayenta, you will have to stand up infrastructure to orchestrate canary deployments shown here by our Phase 1 diagram.

At Nike, we made a pipeline that stands up the infrastructure, calls and polls the standalone endpoint, and generates a report from the canary results

After the canary judgement has been made, if the canary passed, we want it to be deployed to production, and if the canary failed, we want it rolled back. So we also orchestrated the process for promotions and rollbacks. We made manual and automatic modes for the promotions and rollbacks to enable our users to do what works for them. In automatic mode, the canary will be automatically promoted if it passed and rolled back if it failed.

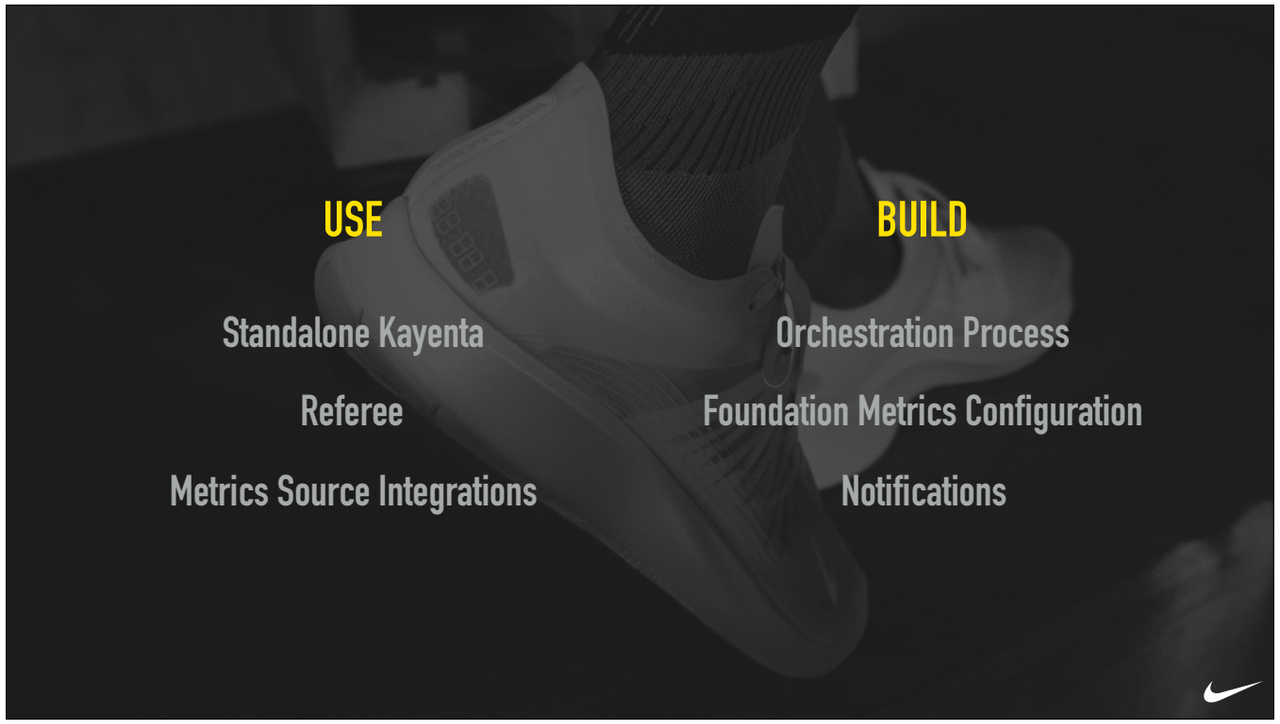
In manual mode, the user will be prompted to make a final decision for promotion or rollback, regardless of whether the canary passed or failed.

You also will have to update your foundation metrics configuration so it reports server group and location metadata



At Nike, we additionally wired up notifications so that users are notified when their canary is complete.

We did this by implementing an event listener in a custom Kayenta plugin.

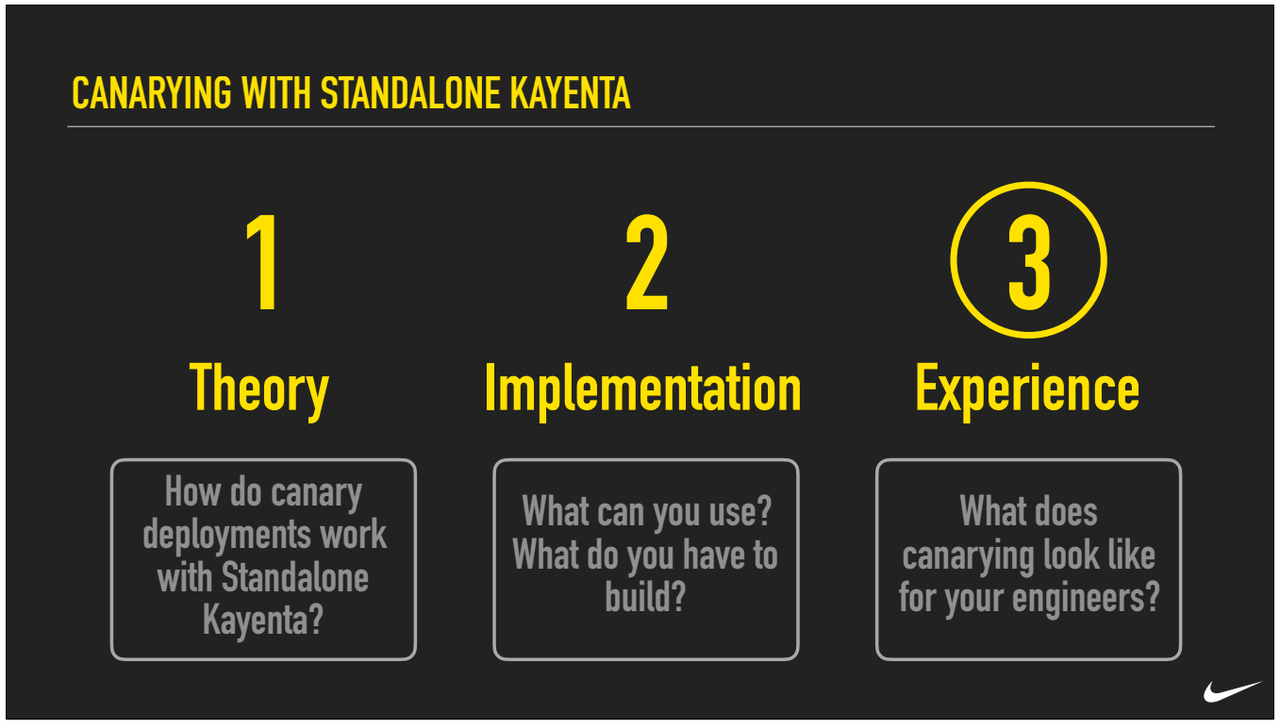


To recap, this is what is available for you to use,
Standalone Kayenta, Referee, and Metric Source Integrations.

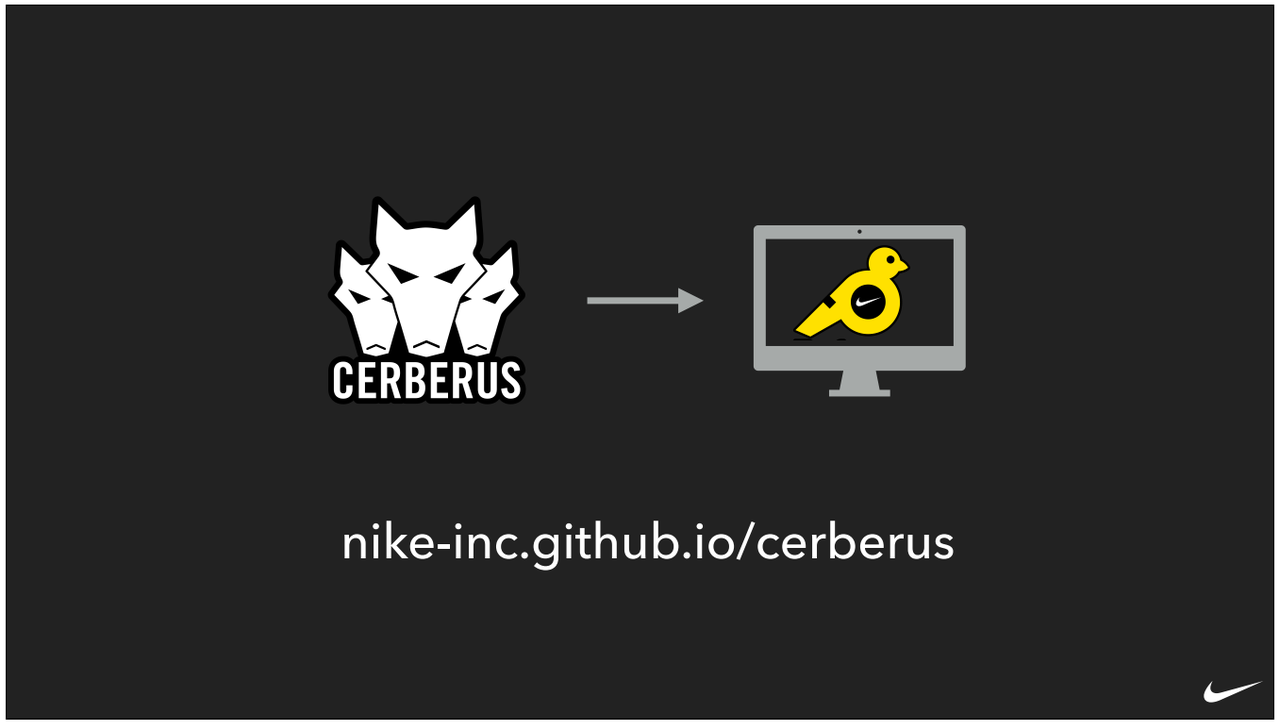
click

and this is what you will have to build.

You have to set up the orchestration process which is Phase 1
You have to update your foundation metrics configuration
And optionally wire up notifications.



We have moved through theory and implementation.
Now experience: what will canarying with Standalone Kayenta actually look like for your engineers?



Let’s just show you our own example.

Working on canarying and Referee isn’t actually our day job.

We work on another product called Cerberus, an **open source** secure property store for cloud applications.

Click and URL drops in

Check us out!

We wanted to use canary analysis with Cerberus, so while we were developing the whole process of using Standalone Kayenta with Referee, we were wiring it up with Cerberus.

We currently use our entire canary analysis process in production with Cerberus.

In order to canary,

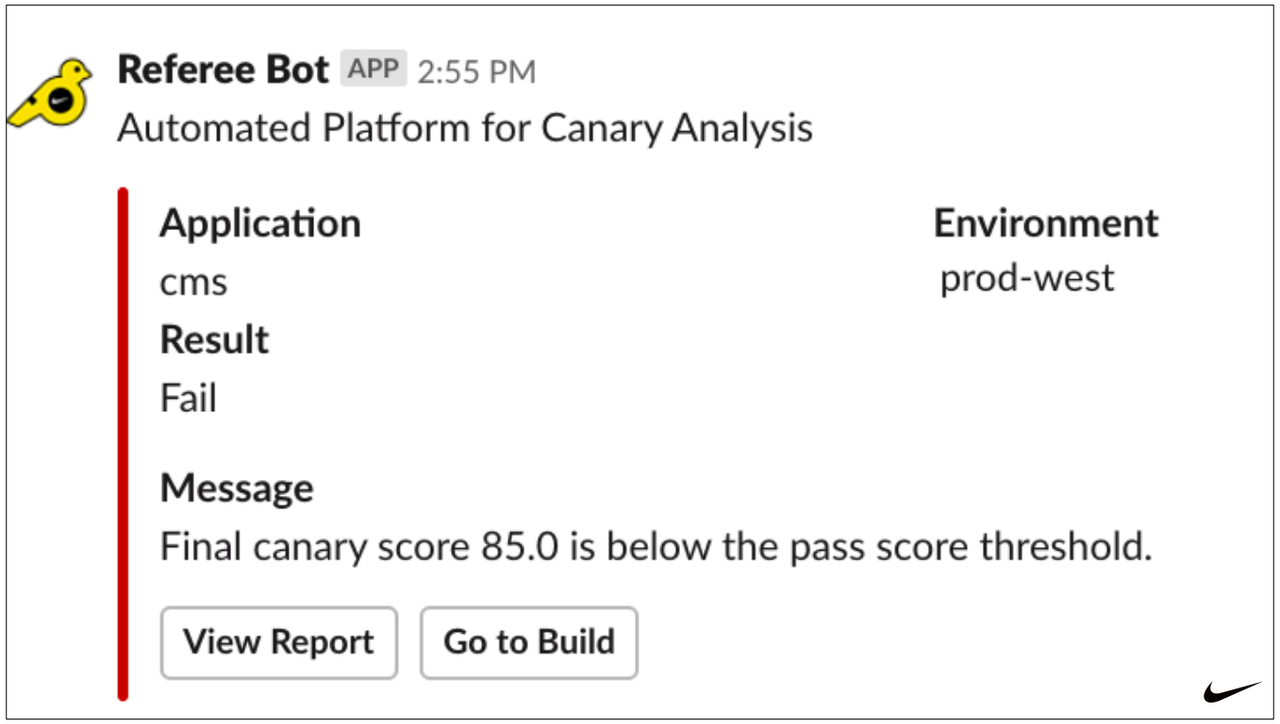
We had to instrument our metrics

Integrate into our pipelines

Create our canary config and test it

And then run canary deployments!

So we had canary deployments working for a couple of months.

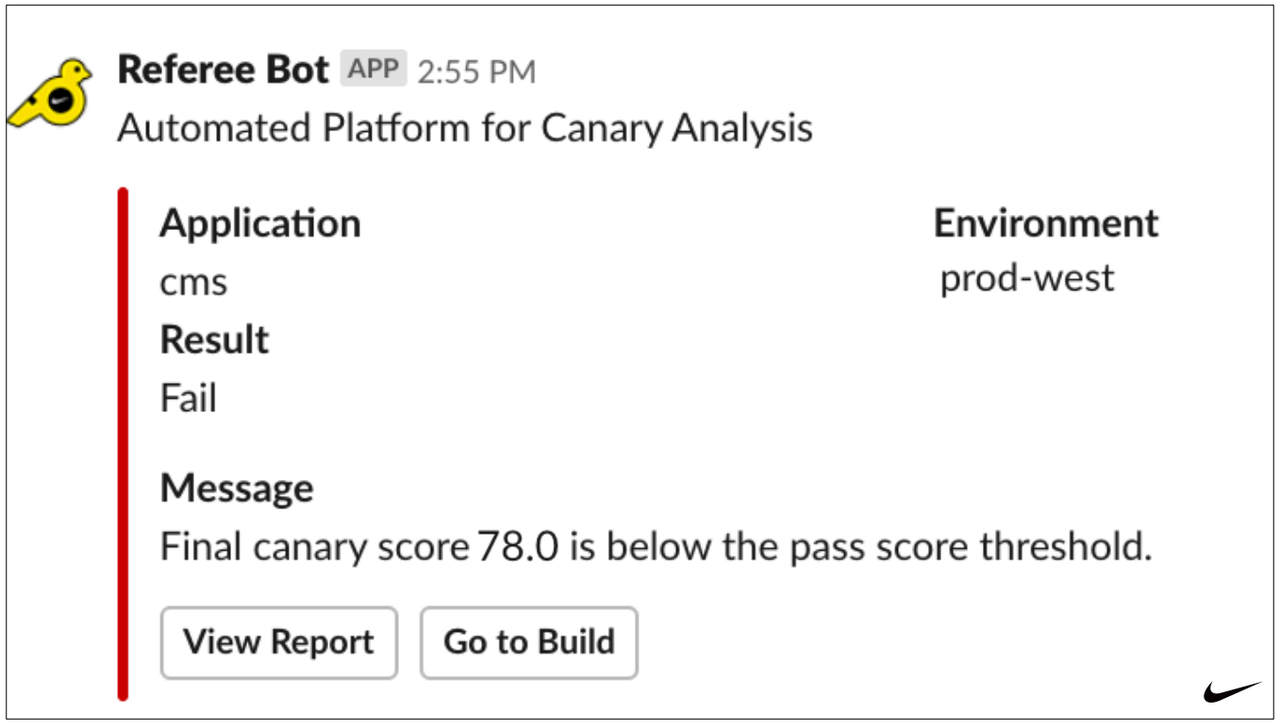


Then one day we got a Slack notification.

Our canary failed!

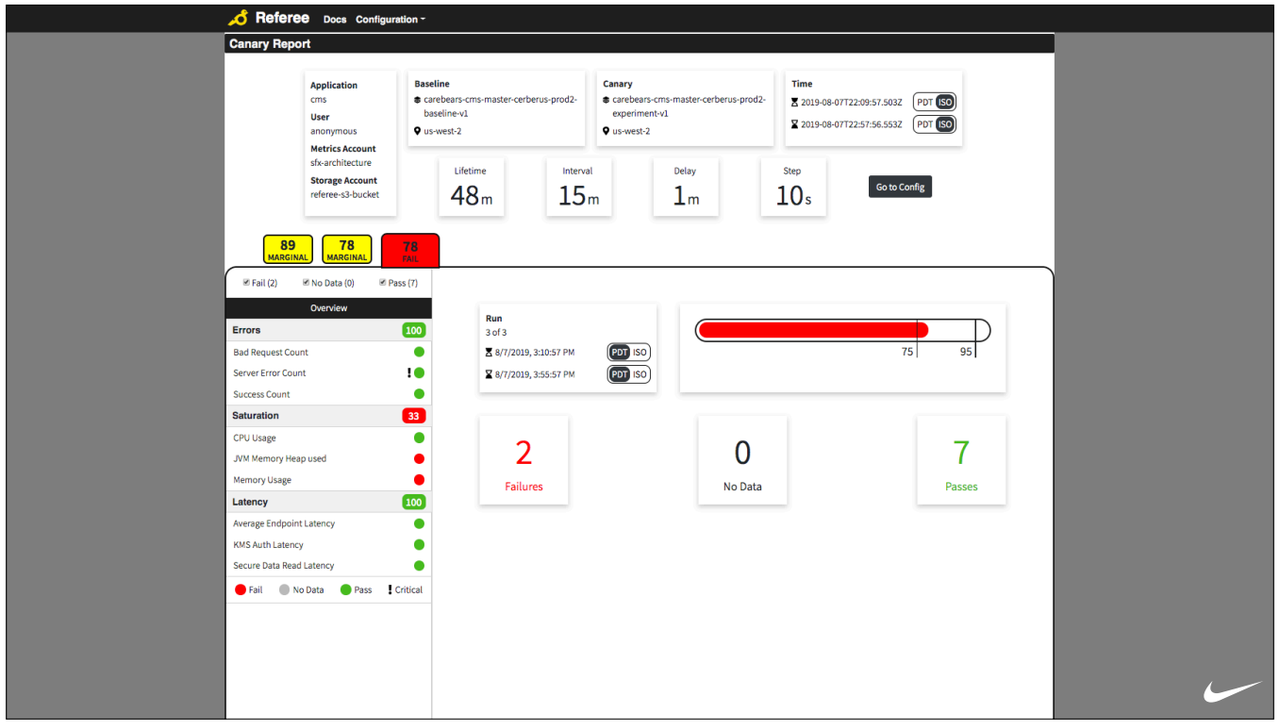
We were confused because we made a fairly innocuous code change.

Now if this is starting to sound familiar, you are absolutely correct. This is the exact same situation we outlined in the beginning of this presentation…..
Because it happened to us!!

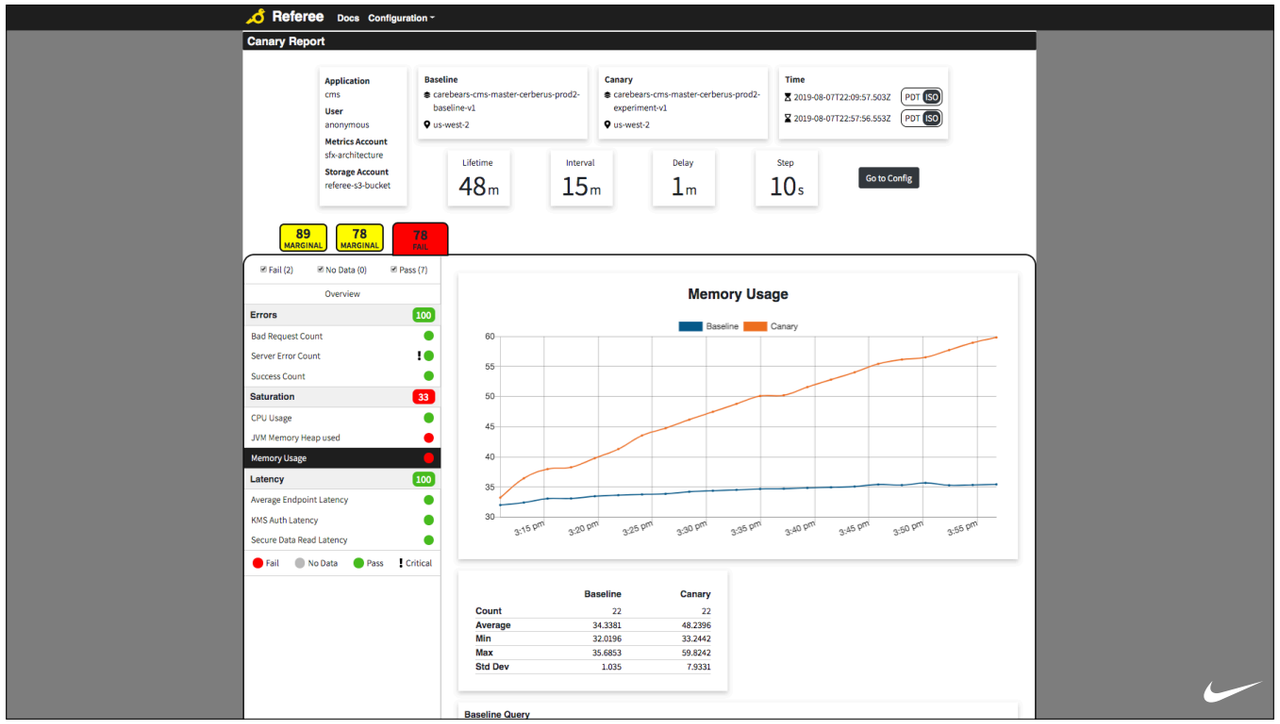


But this time….we had canary

So we clicked on the report



And we investigated the failed canary.



The canary appeared to have a memory leak as we can see here from this graph on Memory Usage, where the canary is represented by the orange line spiking across your screen.

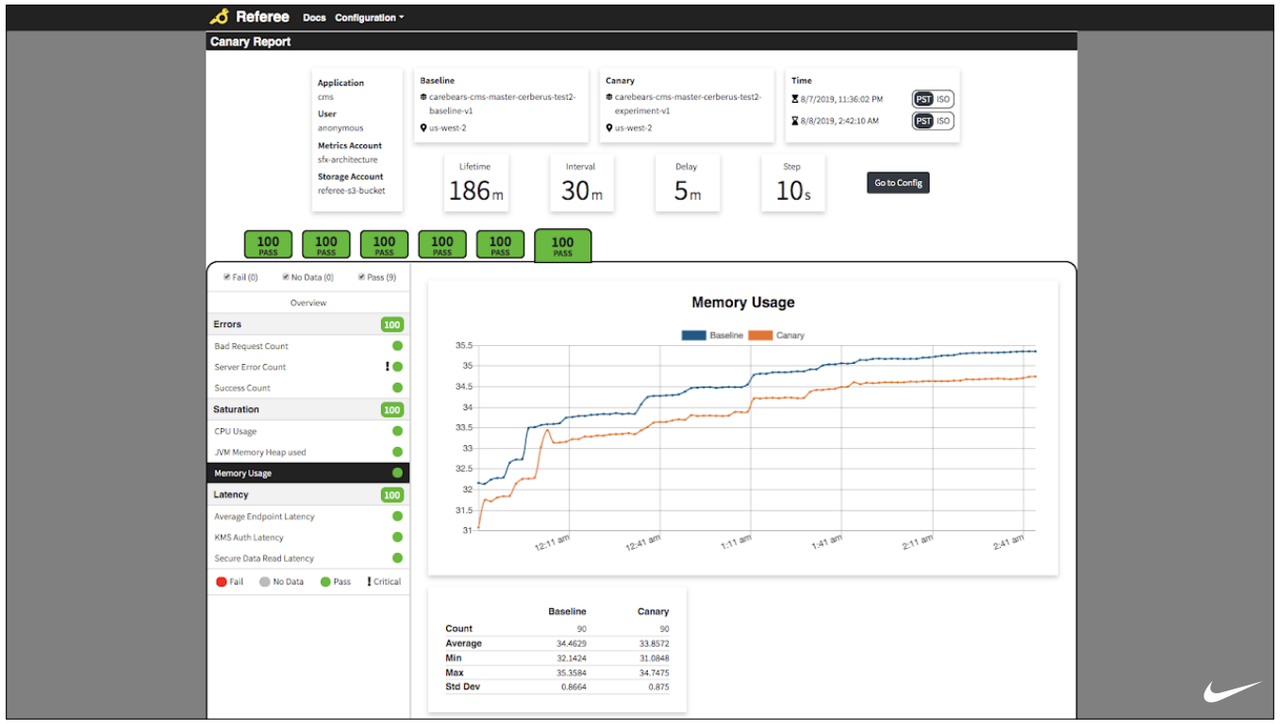
So we thought our memory was going out of control but we had no idea why. The code change we made was minor and shouldn't have affected memory.

We rebaked a new AMI with what was last working in production. We did another canary deployment, expecting the canary to pass because this was on code that had already been working in production.

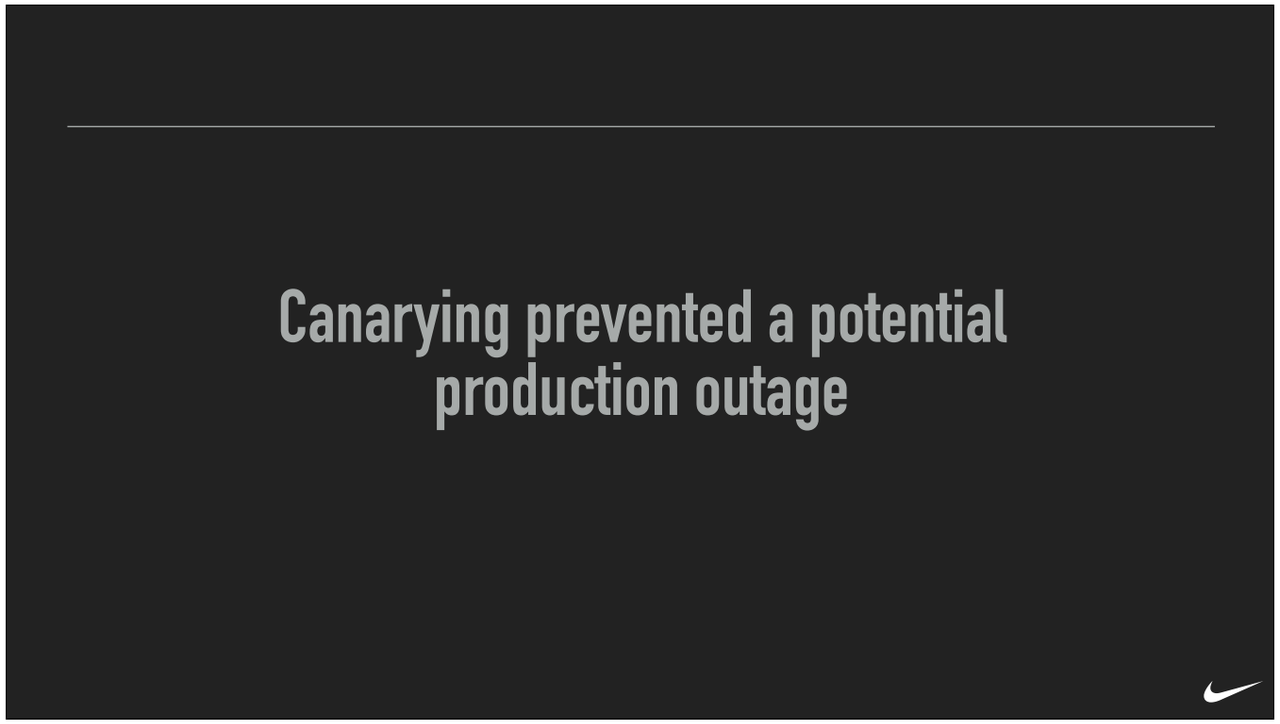
But the canary failed AGAIN. What was going on??!!

We hypothesized that something external had changed so we worked with our foundational infrastructure team to do a root cause analysis. After digging deeper, we realized that a middleware agent on the system had a security update.

Our team rolled that change back and we baked a new AMI with the old agent but the new code.



We reran the canary and the canary passed. We finally could deploy!!



Canarying prevented a potential production outage.

Because we used standalone Kayenta and Referee, this change never fully went out to production and we avoided a large scale memory leak.

Many teams use Cerberus and if we have an outage, that has ripple effects for all of the services that depend on us.

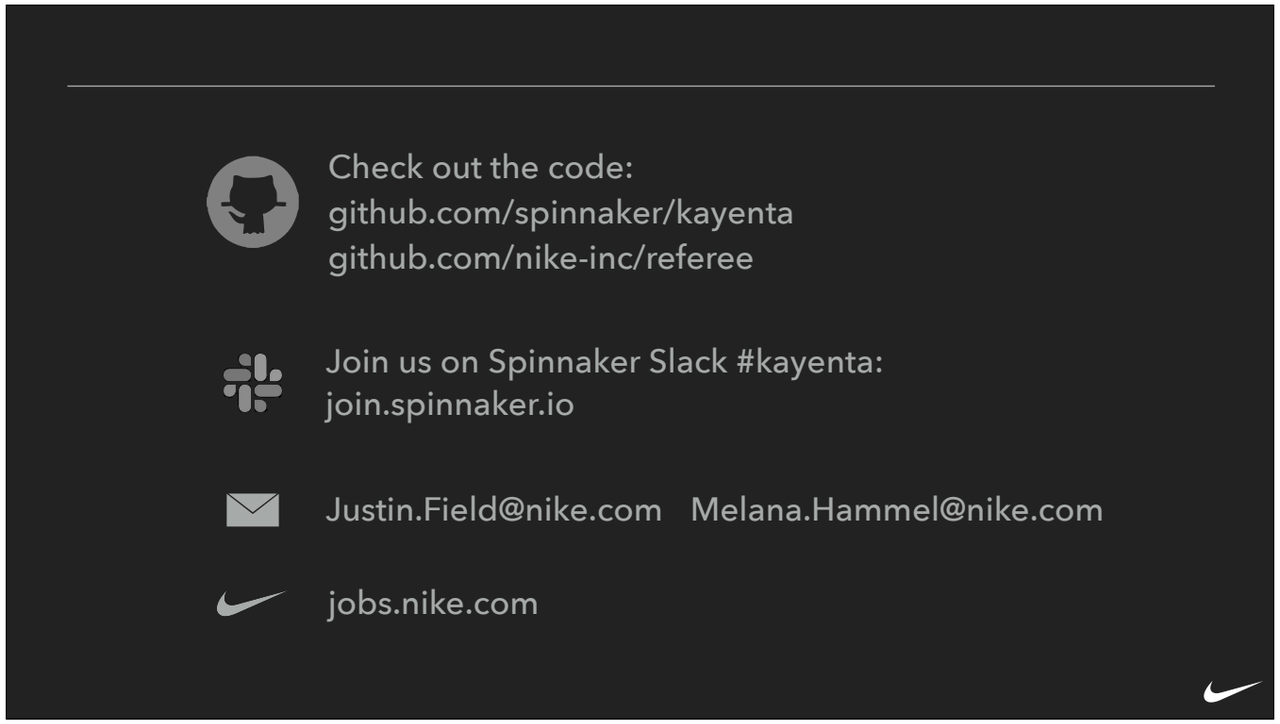
This was our story but it could be yours.

Make your life easier as a developer. Spend less time manually watching graphs and logs. Prevent large scale production issues.

Canary.

Justin is going to walk you through how you can get started today.

So what can you do today?



You can start canarying today using Spinnaker.

If you want to use standalone Kayenta, we have documented a lot of what we discussed today with more technical details as markdown documentation in the Kayenta project, so check that out.

Check out the referee project if you're interested in a UI just for Kayenta.

Join us in the Kayenta channel on Spinnaker Slack, we are active there

Feel free to email us.

If you liked our story, Nike is always looking for awesome engineers.

RESOURCES

- Canary analysis: Lessons learned and best practices from Google and Waze
- Best practices for configuring canary
- 9 Reliability-Based Best Practices for Canary Deploys
- Canarying Well: Lessons Learned from Canarying Large Populations
- Matt Duftler & Michael Graff: Performing Automated Canary Analysis with Kayenta and Spinnaker
- Monitoring Distributed Systems: The Four Golden Signals

Here are some links to materials we found useful during our journey.

The deck should be available online, email us or reach out in Slack if its not.

Thank you….

Pause

Who has the first question?