ManagedKube

About Us

ManagedKube is a DevOps software engineering consultancy that helps companies run large-scale, reliable applications. We work with the entire development team to architect, design, build, optimize, and operate infrastructure in the cloud (AWS and Google Cloud).

We help organizations effectively and efficiently incorporate DevOps principles, resulting in more agile and productive organizations. We also advise companies on how to strategically leverage the cloud in a way that makes sense for them.

Our specialties are Docker, Kubernetes, systems automation, security, and migrating workloads to container-based deployments. We use our technical skills and know-how to help companies build and deploy applications faster and more reliably.

We've worked with over a dozen different clients over the last four years and are proud to say that the majority of our clients come from referrals. We've worked with multiple folks at one company who have then brought us onboard at their next company.

We'll highlight a few of the core competencies of our team and a few case studies below so you can get to know us better.

Additional information on our website:

- https://managedkube.com/devops-consulting
- https://managedkube.com/devops-infrastructure

Linkedin: https://www.linkedin.com/in/garland-kan-a62890/

info@managedkube.com

ManagedKube

Core Competency #1: Team and Knowledge Building

ManagedKube assembles a team with the right expertise to tackle each project. This is largely dependant on the client's internal team and the scope of the project. We have a number of DevOps experts on our roster that we pull in as-needed for different projects.

Because we are a small team, our projects are primarily short-term engagements that consist of understanding what an organization most needs from their infrastructure, building that custom-tailored system, *and* educating our client's team on how to operate and maintain the system. We firmly believe that the key to ManagedKube's success and happy customers is our attitude towards knowledge building. We freely share knowledge and teach what we know so we can develop our client's internal team and set them up for success after ManageKube's contract is over.

In practice, we work closely with our clients in a collaborative manner during architecturing and project implementation to help them gain the operational insight and experience needed to operate the environments. By taking this approach, we avoid a big "hand off" at the end of a project where there is too much information to feasibly absorb. At the end of a project, our clients can confidently operate the systems because they know how the system was built and why decisions were made.

Core Competency #2: Long-term Systems Thinking

ManagedKube takes a long-term view to its work. We show our clients the best techniques and workflows to create the most efficient deployment processes over the lifetime of their software. We have a strong point of view that DevOps teams should invest the time upfront to build infrastructure and processes the right way from day one. This avoids incurring technical debt that you'll pay back many times in the future. You may save time today by building infrastructure in a quick and dirty fashion, but we've seen this go awry too many times to count. We always advocate for the best long-term solution for the entire system – the software and the business.

For example, in legacy IT environments, resources might not be expressed in code. This may be fine when you first deploy something, but later on when you want to rebuild or build a new environment, you'll run into issues because the knowledge on how something was deployed is not readily available. Since infrastructure is the foundation that everything else is built on top of, you want this piece to be one of the most solid and reliable pieces of your stack. We practice infrastructure as code methodologies with all of our clients. All infrastructure pieces should be built with code or configurations that is easily testable and repeatable. This ensures that when anyone wants to make a change in the future, they have all the information needed to do so safely.

We are also big proponents of a Gitflow workflow. With everything expressed as code or config, this lends itself very well to a Gitflow process. We believe infrastructure code and configs should be treated like any other piece of software and go through the same workflow. This usually means branching off for new tastes, automated unit type testing, peer reviews before merging, and automated deployments.

ManagedKube

Core Competency #3: Leveraging Tools

Our philosophy is to leverage publicly available tools as much as possible - why build what has already been built? We customize and integrate the right suite of tools, both open source and proprietary, to meet the specific needs of each client.

This practice saves our clients time and money by dramatically accelerating a project's progress. Software development has evolved to a point where infrastructure code is becoming more and more commoditized. While software has a long way to go before it's 100% turnkey, we leverage the available infrastructure building blocks as much as possible.

Our expertise lies in knowing what tools are available, the best tools to use, their benefits and tradeoffs, and how they integrate with each other. Since we specialize in the infrastructure automation and scaling space, we are well versed with what is out there and how each piece works together.

Half the battle is knowing what tools are available - ManagedKube dedicates time to keeping up-to-date on the tool landscape by constantly reading the latest DevOps blogs, attending meet-ups, and being asked to evaluate the latest tools due to the nature of our consulting work.

Core Competency #4: Working with a Diverse Client Base

We've worked with a wide range of clients in a variety of ways:

- from 6 employees to a Fortune 500 company
- from advertising technology, to biotech, to information security
- from a project for a few weeks to a multi-year engagement

This means that we understand what works, where. This hard-won experience and priceless knowledge helps us quickly identify how to approach each client's unique problem. The right solution for a small company is likely not the right solution for a big company, just like the right level of infrastructure automation is different for each company's product. ManagedKube is experienced at building scalable infrastructure for a wide range of companies and industries.

We are able to apply learnings from one client to all of our clients. With our guidance, our clients benefit from avoiding the pitfalls of all of our clients before them. We are able to avoid tools that aren't ready for showtime yet or quickly implement tools across all of our clients that solve a pressing pain point. ManagedKube knows what works in the real world; as the ecosystem around Kubernetes becomes more crowded, we are here to help you avoid obstacles and speed bumps in your cloud technology journey.

ManagedKube

Client Case Studies

Lucidworks

The Company: LucidWorks is the maintainer of the popular open source software Solr. They traditionally have sold a packaged enterprise solution of the Solr software.

The Problem: LucidWorks' customers were increasingly asking for a hosted (SaaS) solution of the software because Solr is not easy to operate and scale. Their internal team had AWS experience from using instances for development purposes but did not have experience running and operating a production SaaS solution.

Our Solution: ManagedKube helped LucidWorks build a SaaS solution *on the cloud* in less than 8 months start-to-finish with a Docker/Kubernetes system in 2017 (just 18 months after Kubernetes was launches)

Our Approach: We collaboratively worked with LucidWorks' management and development teams to map out the problem, establish what success would look like, and to ultimately build the best solution to achieve that end result. We answered key questions about how to build the infrastructure, such as:

- Do we go with a configuration management tool such as Chef, Puppet, or Ansible?
- Do we use Cloudformation or Terraform to build the infrastructure?
- What OS should we use?
- What does the development life cycle look like?

The most critical question that we helped LucidWorks tackle was, should Lucidworks build an infrastructure model with a configuration management base or Kubernetes? In 2017, there were still a lot of folks who believed that using configuration management to create programmatic infrastructure as code was the best solution. However, we firmly believed that containers and Kubernetes are a better way of creating and managing infrastructure (which the passage of time has proved out) and were able to guide Lucidworks to building a highly scalable infrastructure on AWS.

GUARDANT HEALTH®

The Company: GuardantHealth is a public biotech company that develops blood tests for early detection in high-risk populations and recurrence monitoring in cancer survivors.

The Problem: GuardantHealth's compute operations were entirely on-premise when ManagedKube started working with them in 2016. They wanted to explore what moving to the cloud would mean for them, especially considering HIPAA privacy requirements. ManagedKube worked with them to move their genome sequencing pipeline to run in the cloud as a prototype for the rest of their compute.

Our Solution:

1. We designed the process and led the implementation team to copy local genome sequencing data to AWS via AWS Snowball and over the internet (700TB of total data moved at 2TB/day), taking great care to make these data transfers secure and reliable.
2. We designed and built infrastructure with:
    - Multi-region data at rest design with rules to age data out to lower tiered storage to make holding this amount of data cost effective

- Automation constructs on how specific data can be retrieved and loaded into an AWS compute environment to re-run the data pipeline on it
- A Kubernetes platform for running web type workloads for development and production environments on AWS.
- Full monitoring, logging, and visibility packages
- A fully automated CI/CD pipeline to build and test the software, containerize it, perform integration tests, and a deployment sequence into: dev, qa, staging, and production.

Our Approach:
The ManagedKube team worked with GuardantHealth to understand how the genome sequencing pipeline worked end-to-end, what steps were involved, and how to validate a successful run. With this knowledge, we created a pipeline that would run in AWS. We showed various configurations of the pipeline running on bigger/smaller machines and even Spot Instances.

The Company: Tillster creates online and mobile ordering systems for companies such as Kentucky Fried Chicken and Jollibee. They are responsible for these companies' back end infrastructure, which often interfaced with local stores to get their menu and pricing. These systems also processed credit cards which means they are subject to PCI level 2 compliance.

The Problem: Tillster needed help transforming their development workflow and systems. Deployment of new code was time consuming, involved many people, and was typically performed at off hours, which the team disliked. Plus, their infrastructure was hard to maintain.

Our Solution: ManagedKube shifted a lot of their tenants over to the new containerized system and paved the way for shifting the rest of their tenants over to the new system. We are happy to say that the Tillster team has fully taken over operating and maintenance of the system that we built with them.

Our Approach: ManagedKube talked to all of the stakeholders (management, program managers, and developers) at Tillster to understand everyone's pain points. We took this information and created a plan for how to move forward with a next generation development workflow and containerized platform. After discussion and revisions with the Tillster team, we led the implementation of this plan. We slotted fully into their Agile sprint cycles, attending scrum meetings like full-time employees, to implement and deploy the new solution.

The most critical question that we helped Tillster answer is, how should they migrate their current application to a container running on Kubernetes? Should they start from an empty container and build it back from the ground up or should they work from the current Chef builds?

We recommended that they take an entirely built instance of the application and copy this directly into a container to launch onto Kubernetes. This allowed us to do two things: quickly test how and if their workload would work on Kubernetes and a mostly finished product where we can build an automated build and deploy pipeline. While this was not efficient and not a product you would run in production, it did give us a development workflow where we had a testable object with very defined outputs and a process to start changing this application. With this workflow in place, the team started to pull apart the container and splitting out parts that had natural separations into microservices. They also pulled out the secrets and configurations

needed to make the containers portable. Lastly, the team refactored the artifacts built in the container into other pipelines to make the whole process reproducible. This approach allowed multiple people to work on different pieces of the application at the same time without affecting each other. They could then integrate each piece back in when it was completed and tested. As a result of this strategy, we were able to refactor the image we copied into the container image and separate it out into multiple microservices, with everything reproducible via code and in an automated pipeline.

A key characteristic of the system we built for Tillster was a CI/CD pipeline that abstracted away most of the infrastructure and deployment from their developers. This set-up simplifies the deployment process for software developers because they only need to interface with Git and Jenkins. They do not need to interface with Kubernetes directly. Their workflow starts with Git, where they make changes to the versions of the software they want to build and deploy. Once that change has been committed and pushed into a branch, the developer can give that information to Jenkins for building and deploying. Tillster has a complex multistep build pipeline, sourcing artifacts from various places and building multiple containers for a deployment. Jenkins automates this complexity with one build config that the developer has full control of. Once the artifacts such as the containers are built and tested, the developer has options in the Jenkins GUI to deploy this set of artifacts to an environment (Kubernetes cluster). From there, Jenkins will run automated tests against that environment to ensure it is working as expected.