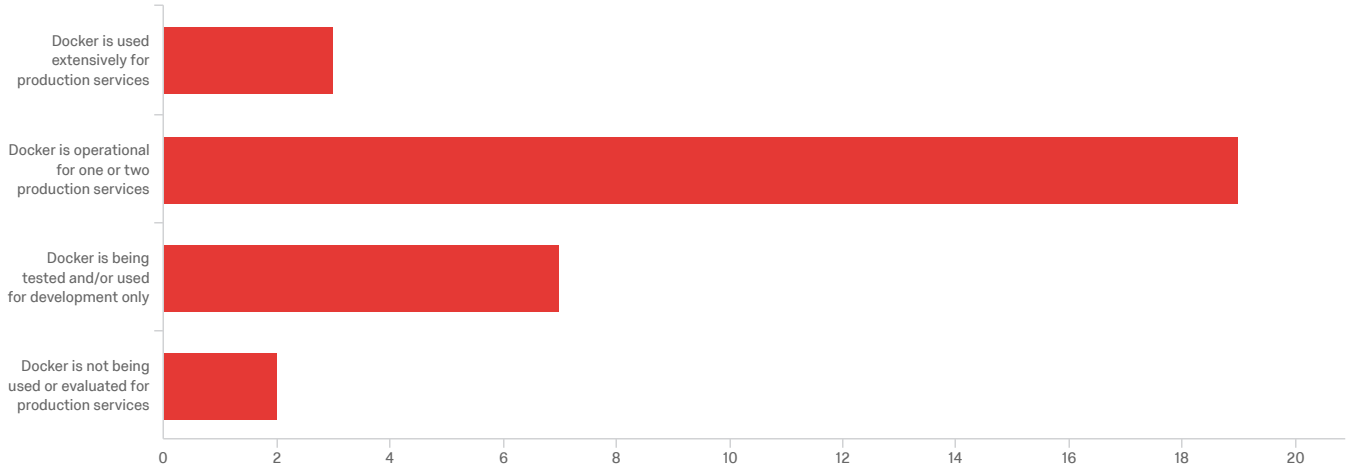
Default Report

TIER Container Orchestration

September 10, 2018 9:35 AM MDT

Q2 - Summarize your organization's current use of Docker containers. By "your organization", we mean your area (e.g., central IT) as opposed to your school as a whole.

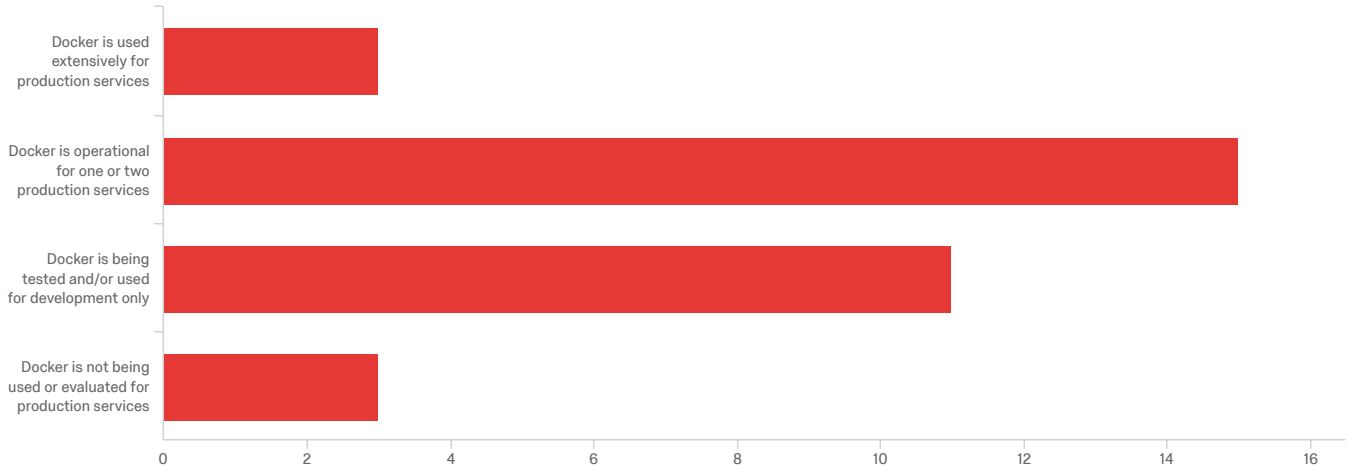


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Summarize your organization's current use of Docker containers. By "your organization", we mean your area (e.g., central IT) as opposed to your school as a whole.	1.00	4.00	2.26	0.72	0.51	31

#	Field		Choice Count
1	Docker is used extensively for production services	9.68%	3
2	Docker is operational for one or two production services	61.29%	19
3	Docker is being tested and/or used for development only	22.58%	7
4	Docker is not being used or evaluated for production services	6.45%	2
			31

Showing Rows: 1 - 5 Of 5

Q5 - Summarize your group's current use of Docker containers. By "your group", we mean your area (e.g., IAM Services) as opposed to your school or organization as a whole.

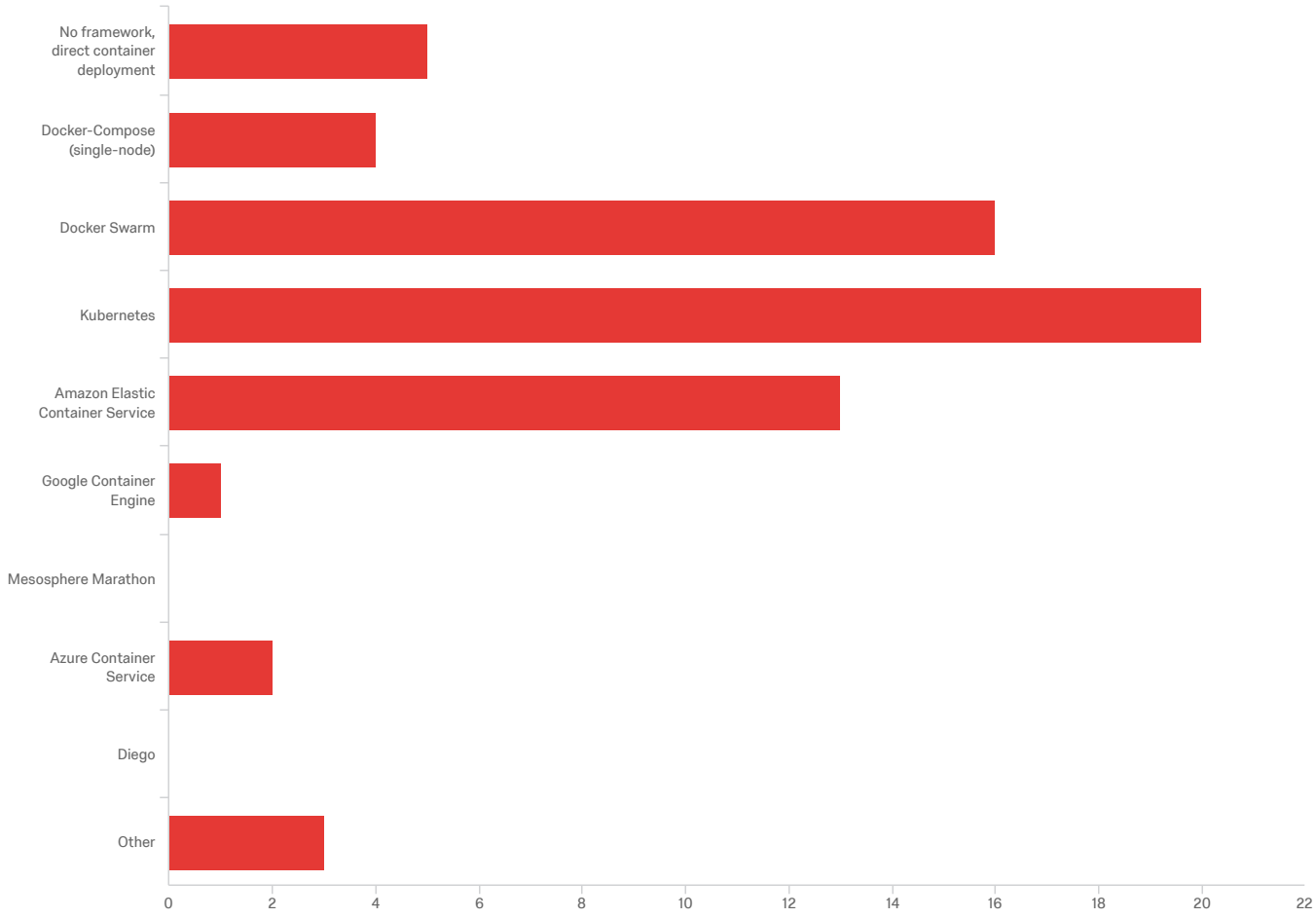


#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Summarize your group's current use of Docker containers. By "your group", we mean your area (e.g., IAM Services) as opposed to your school or organization as a whole.	1.00	4.00	2.44	0.79	0.62	32

#	Field		Choice Count
1	Docker is used extensively for production services	9.38%	3
2	Docker is operational for one or two production services	46.88%	15
3	Docker is being tested and/or used for development only	34.38%	11
4	Docker is not being used or evaluated for production services	9.38%	3
			32

Showing Rows: 1 - 5 Of 5

Q6 - Based on your knowledge of Docker and your organization's direction, which orchestration framework(s) are you most likely to need to use?



#	Field	Choice Count	
1	No framework, direct container deployment	7.81%	5
2	Docker-Compose (single-node)	6.25%	4
3	Docker Swarm	25.00%	16
4	Kubernetes	31.25%	20
5	Amazon Elastic Container Service	20.31%	13
6	Google Container Engine	1.56%	1
7	Mesosphere Marathon	0.00%	0
8	Azure Container Service	3.13%	2
9	Diego	0.00%	0

10	Other	4.69%	3
			64

Showing Rows: 1 - 11 Of 11

Other

Other
dunno
Amazon Elastic Container Service for Kubernetes Amazon (EKS)
RedHat OpenShift

Showing records 1 - 3 of 3

Q7 - Your expectation of how hard it would be to leverage a TIER Swarm docker-compose file to assist with your container orchestration needs?

#	Field	Minimum	Maximum	Mean	Std Deviation	Variance	Count
1	Convert file for use with your campus framework	0.00	75.00	35.48	20.56	422.53	29
2	Read file and use as a reference while creating campus configuration	0.00	72.00	36.36	18.40	338.47	25

Q8 - What do you see as the main set of trade-offs between Kubernetes and Docker Swarm?

What do you see as the main set of trade-offs between Kubernetes and Docker...

Kubernetes is certainly more complicated, but more robust, complete and undergoing *much* more development.

The ecosystem still hasn't made up its mind on which model that they will support moving forward.

Kubernetes appears to be gaining acceptance and adoption across many vendors. We want a solution that works with our cloud vendor exit strategy seamlessly, but is not a SPF itself, or a vendor lock itself. Docker provides mobility across public cloud providers, but is itself a vendor with a proprietary solution. So while good, it's not ideal. I personally like AWS ECS and Google's GCE, and I don't have as much of a fear of cloud vendo lock-in (when we're not even really "in" yet). But a reasonable plan for future agility is a good thing.

Swarm is tightly integrated with Docker itself and seems to be simpler to set-up, especially for shops that do not already have a large container infrastructure. Kubernetes seems to be more powerful and have a larger community base, but at the expense of more difficult set-up and configuration.

We run everything in Amazon and Kubernetes is a managed service. Any multi-cloud plans we have also involve the managed kubernetes services offered by Google or Microsoft.

Kubernetes is extremely replete with functionality but is quite complex to use, initially. Swarm is dead simple but is very limited i.e. we would need to supplement Swarm with other infrastructure as code tools/products.

Ease of deploying in our local infrastructure (swarm). I don't think we can make the staff commitment to roll our own Kubernetes infrastructure. If deploying to a PaaS solution; we would definitely look at Kubernetes.

Since UNCG is in the earliest stages of container deployment, it's utilization is rather low (<10 production containers) and limited to non-orchestrated deployments. Our infrastructure architecture group has done some initial exploratory projects and it looks like Azure Kubernetes Services is our front-runner for cloud-based container orchestration. A lot of the flexibility that AKS offers "out of the box" vs. an on-prem setup seems to be an attractive attribute. Given our desire to develop a cloud-first strategy, I think a significant amount of effort will be made in this direction going forward. When it comes to on-prem orchestration, however, given Kubernetes' additional setup/configuration, it becomes more of a toss-up.

Kubernetes is much more comprehensive but difficult to manage locally without significant staff training. Docker Swarm is simpler but with less functionality. Probably the Kubernetes "in the cloud" solution is likely to be a sweet spot.

Kubernetes is where all the major IaaS platforms are headed long-term. It has support from the CNCF.

Docker Swarm configuration is less complex than Kubernetes awarding it some advantages when getting started in containerization. It also scales well with relatively small effort. Kubernetes, though more complex, has better monitoring and logging capabilities up front. Institutionally, we use Docker Swarm, but have upgraded Docker Enterprise so that we can begin evaluating Kubernetes in Docker.

Our main reason for using Swarm is the ability to use compose files, which is what we are already doing to manage multi-container development environments. We have not had much cause to experiement with or learn about Kubernetes yet.

1. Kubernetes has more active development and more large organizations providing development input/hours. 2. Kubernetes has built-in load balancing capabilities that the community edition of Docker Swarm does not 3. Kubernetes has some additional built-in access controls.

From a technical standpoint, I can't say because I don't know much about Swarm. But from a pragmatic standpoint, I rarely hear or read about Swarm outside of the Tier channels. Everybody is talking about ECS or Kubernetes, when they're not talking about how containers themselves are obsolete and everybody should be going Serverless now.

Lower barrier to entry with Docker Swarm. And given how resistant some areas of the organization are to Docker in general, the lower the barrier to entry the better.

Unfortunately I am not educated enough to know. Docker is more "known" and folks around campus are getting used to the concept of containers. No one I talk to on campus has any knowledge of Kubernetes or understands the differences/benefit it would have over Docker.

unsure, have not used Kubernetes

Docker Swarm- reduced initial complexity. Loosing market share. Kuberneties - more complex scripts/configs, looks like more sustainable, open-source, long-term approach.

I believe that the container publishing should maybe give a not to detailed example of a docker run command. The container should be versatile and standard enough where it doesn't matter which orchestration platform you are using.

I feel a "barrier to entry" with Kubernetes since it seems the installation is a bit more complex, and there is a CLI separate from Docker (unlike Docker Swarm). But I recognize its emergence as a leader in the orchestration field. If I had to pick one, it would be Docker Swarm. For Amazon ECS, we are using a variation of Docker Swarm to compose and deploy images before launching ECS tasks.

I'm not the technical arms and legs to work with these solutions, but from my understanding, Docker Swarm is losing steam and popularity. Kubernetes is the "in" solution. With that said, my understanding is that Docker Swarm is relatively easy to use which Kubernetes has a steep learning curve.

Kubernetes is a CNCF project, Docker Swarm is not. No need to say more. Providing docker-compose files is of little value, instead a Helm Chart would great. Invest in CNCF projects where possible.

Not enough experience with either to say.

I have not put much effort in Kubernetes. TIER suggested Docker Swarm, given that we had not done any work with containers we decided to go that direction. At some point, we expect that we may move to Amazon or to some other cloud service, but there are no firm plans at this time. I expect that some time in the next 6 months, midPoint, Grouper and Shibboleth will be move into production on Docker Swarm.

Neither offer functionally or cost/benefit justifications for immediate support for locally developed or delivered products. Long term, we will end up supporting them as the industry shifts from rpm/tar deployments to containers. Dockers/Kubs offer a deployment path for upstream developers, but only locally solve scale problems we don't have. Deploying a docker infrastructure for 2 servers is a bit overkill. As long as tar.gz options are available, we will be using those to move our own custom modifications into TIER products and deploying as stand-alone applications. The cost/benefit just isn't there.

Kubernetes positions us to take advantage of different operational environments. We began this process on Docker Swarm and we are already investigating Rancher 2 and Kubernetes to allow for the management of local and remote (AWS) environments. If Rancher 2 had been an option when we started we probably would have bypassed the Docker Swarm setup and moved directly to Kubernetes. The learning curve for Kubernetes is higher than that of Docker Swarm but the architectural possibilities are greater.

kube is one word instead of two.

Fault tolerance, I suppose - but this feels like more of a religious war than a technical one. There's nothing wrong with Swarm but it's not the direction we've chosen. Short term, Kubernetes. Long term, Cloud Foundry/BOSH will probably dictate.

End of Report