Tech Mahindra

Blueprint to Cloud Native Transformation

Path to Drive Business Benefits and Breakthrough Digital Experiences

WHITEPAPER

Connected World.
Connected Experiences.

Cloud Native Engineering (CNE)

Cloud native is an ever-prevalent concept in modern enterprise IT. Used to describe container-based environments that exist in the cloud, CNE is more than an infrastructure choice. It's a transformative approach that is intended to reduce cost, decouple applications, provide real time data insights, accelerate release engineering, improve security, and customer experience. If you're looking for a cloud-native transformation, be aware that the process is complex but well worth the effort. So, let's explore the roadmap for this transformation and dive deeper into the CNE operating model.

Why CNE?

Many organizations have taken steps toward "cloud-based" infrastructure, but cloud native (CN) goes a step further. More than defining what's the definition of cloud native engineering let's talk about the benefits that CNE brings:

- Hybrid infra and release practices to scale distributed systems
- Decoupled application architecture to support distributed systems
- Improved developer experiences
- Native omnichannel experience

Cloud-native applications are built to scale thanks to the highly flexible microservices architecture behind them, which enables upgrades to take place for one component at a time and, thereby, reduces downtime.

Additional reasons to choose CNE include:

Scalability: Cloud native means applications adapt and scale dynamically, allowing them to support a huge number of users, highly distributed internal teams, and extremely large development projects.

Key takeaways

User Interface: Cloud native applications are able to support highly intuitive and functional user interfaces by taking advantage of a range of backend capabilities, like push notifications, customer identification, ios/android toolkits, and more.

Automated: Cloud native applications are always automated to the most optimal level, primarily due to the fact that CN apps need to have very few manual tasks and need to be completely removed from the underlying infrastructure to enable their scalability.

Continuous: By nature, cloud native applications support both continuous integration and continuous delivery (CI/CD). The application development and deployment processes are highly automated thanks to this characteristic.

Schema on Read: Schema on Read (SoR) is an innovative method of handling data, and the need for CN apps to work with loosely structured data enables the use of SoR methods. This gives applications more power for digesting data, and they can do so much more quickly.

API-Centric: APIs are a powerful tool that developers use to enable communication across systems. Cloud native applications use APIs in various ways to provision, deploy, and manage CN services.

As for your organization, adopting cloud native engineering means a whole new operating model, whether you're moving from a cloud-based or a completely on-premises solution. CNE represents new challenges but many more opportunities and near limitless potential for cost optimization, reducing time to market, improving security, and building agility into your organization's core.

The Cloud Native Engineering Infrastructure Operating Model

When adopting CNE, organizations must realize the major changes in the operating model and overall landscape. While complex and with many layers, the CNE infrastructure operating model can be most simplistically broken down into the following components:

Provisioning: The provisioning layer of the operating model consists of all the tools necessary to create and harden the foundation for cloud native applications. Examples include security and compliance frameworks; key management solutions for encrypted files; and automated configuration tooling to build computing environments.

Runtime: The runtime layer has no strict definition, but for practical purposes, it represents any tool that a cloud native application needs to run properly. For cloud native landscapes, runtime focuses on the components necessary for containerized applications to remember and communicate. Think virtual disks, persistence, and security considerations.

Orchestration and Management: The next layer of cloud native requires engineers to determine how they're going to orchestrate and manage applications. This layer focuses on how containerized components of an app are managed as part of a group, which means identifying all services that must connect and communicate with each other. To be truly scalable, cloud native apps require resilience and automation at this layer.

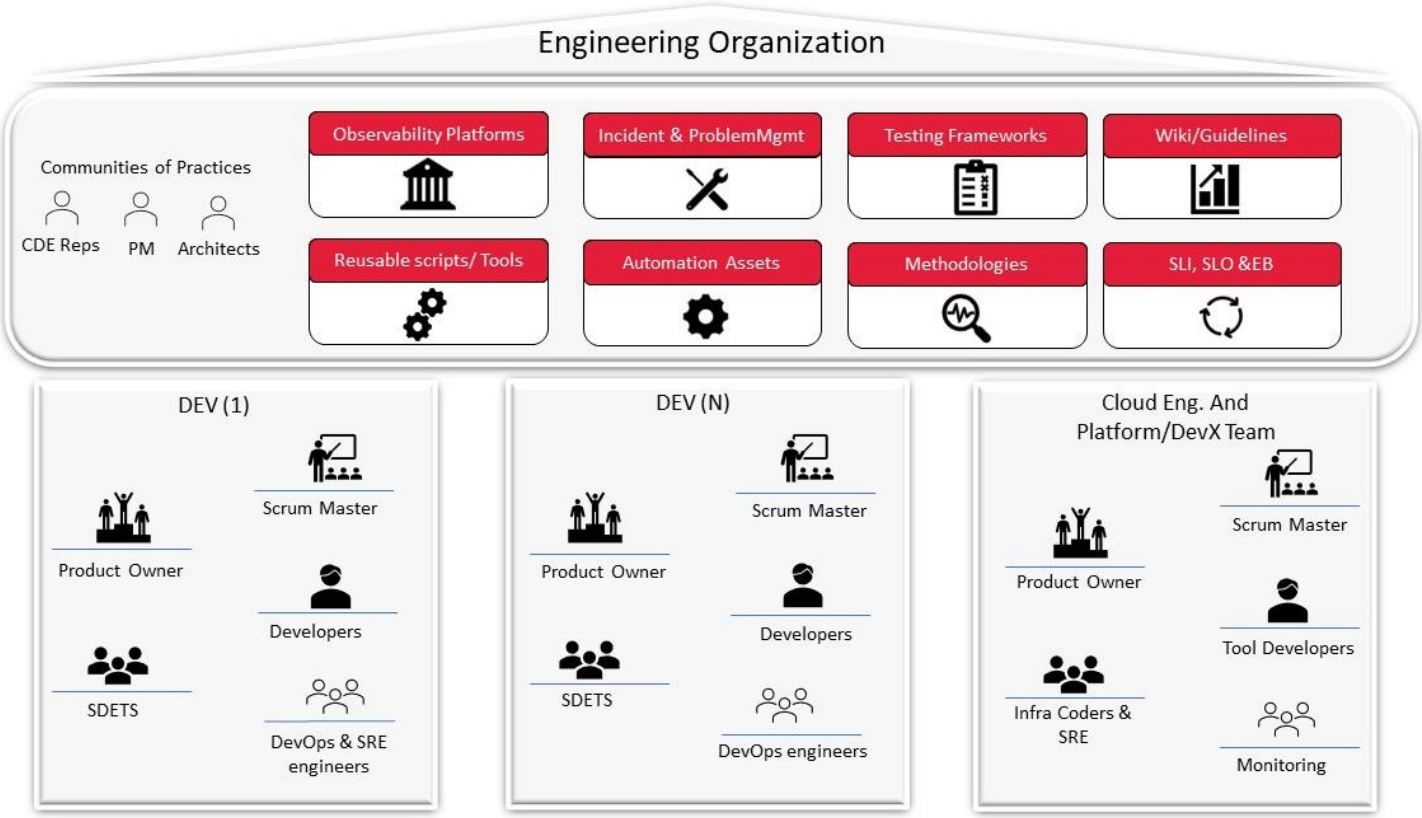
Application Definition and Development: The top layer of the cloud native operation model is the definition layer where engineers actually build their applications. This layer consists of data collection tools, message queues, and services that help engineers configure and maintain their container images to keep apps running.

In addition to these four layers, each one being a prerequisite of the next, the cloud native operating model also requires observability and analysis tools and site reliability engineering (SRE) across each layer to function optimally in a reliable and be fully visible path.

Roadmap for Cloud Native Transformation

There's no "one-size-fits-all" roadmap or strategy that an organization can follow during its CNE transformation. Creating a Center of Excellence (CCoE) is a good idea, where your CCoE team will consist of IT professionals already in your organization who will take on the responsibilities of researching, planning, and guiding the transformation process through change management and extensive communication between departments and teams. The primary goal is to get IT operation team on board with the changes necessary to ensure successful adoption.

Cloud native transformation means shifting focus from infrastructure to actually delivering real-world value and impact through new processes and automation. Lastly, your organization's developers have to standardize the platforms and services they use to support the cloud native approach through the adoption of cloud native app principles and microservices architecture. All of this takes extensive planning, which is why integrated communities of practice teams are crucial.



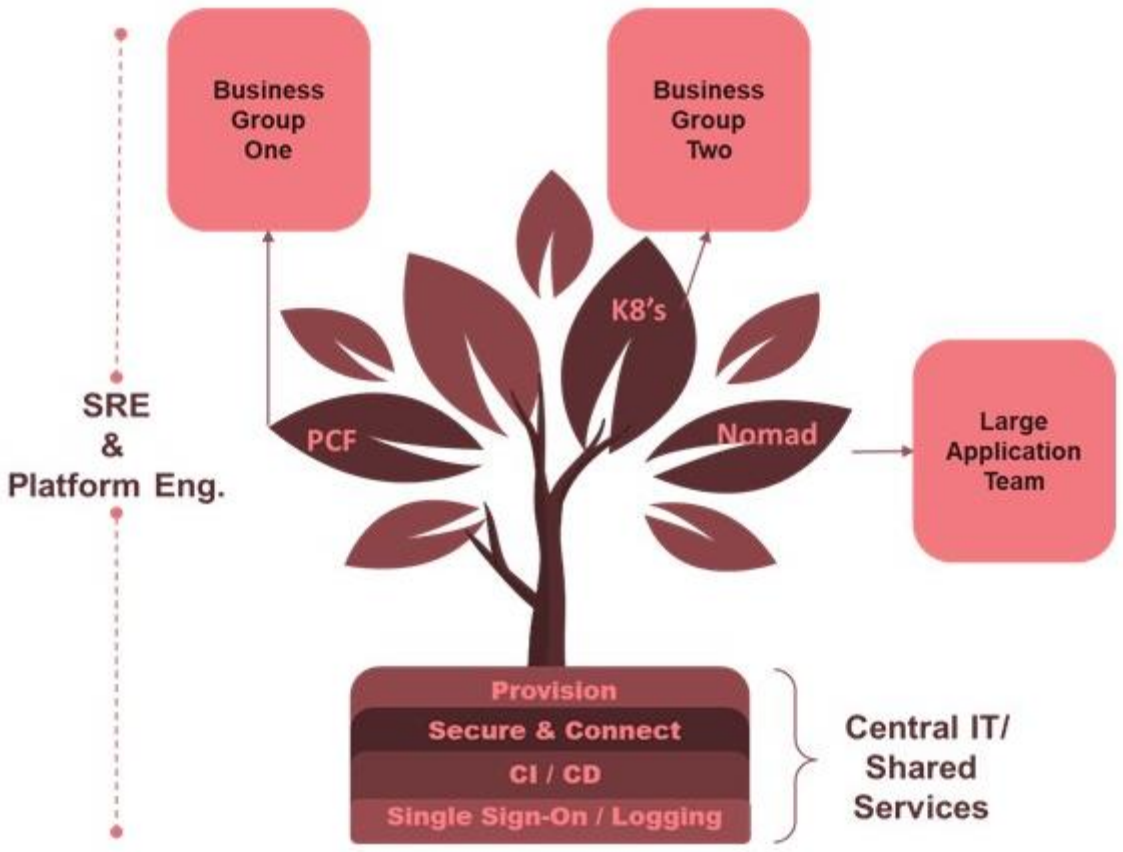
Organization Transformation

As an organization, you need to review your organization's current state to determine whether you currently have a federated IT department or a centralized IT department.

Federated IT: More flexibility, less connectedness. According to UCSB, "In a federated model, departments have more influence on large-scale IT through systematic participation in governance. A federated IT model creates an environment where both autonomous departmental IT units and centralized IT collaborate to meet the needs of the whole."

Centralized IT: Less flexibility, more standardized. According to TechTarget, "centralization is the consolidation of an organization's technology resources, when done correctly, IT centralization can not only simplify administrative tasks, it can improve security, make data management easier and save the company money."

When transitioning to cloud native engineering, most organizations take a middle road.



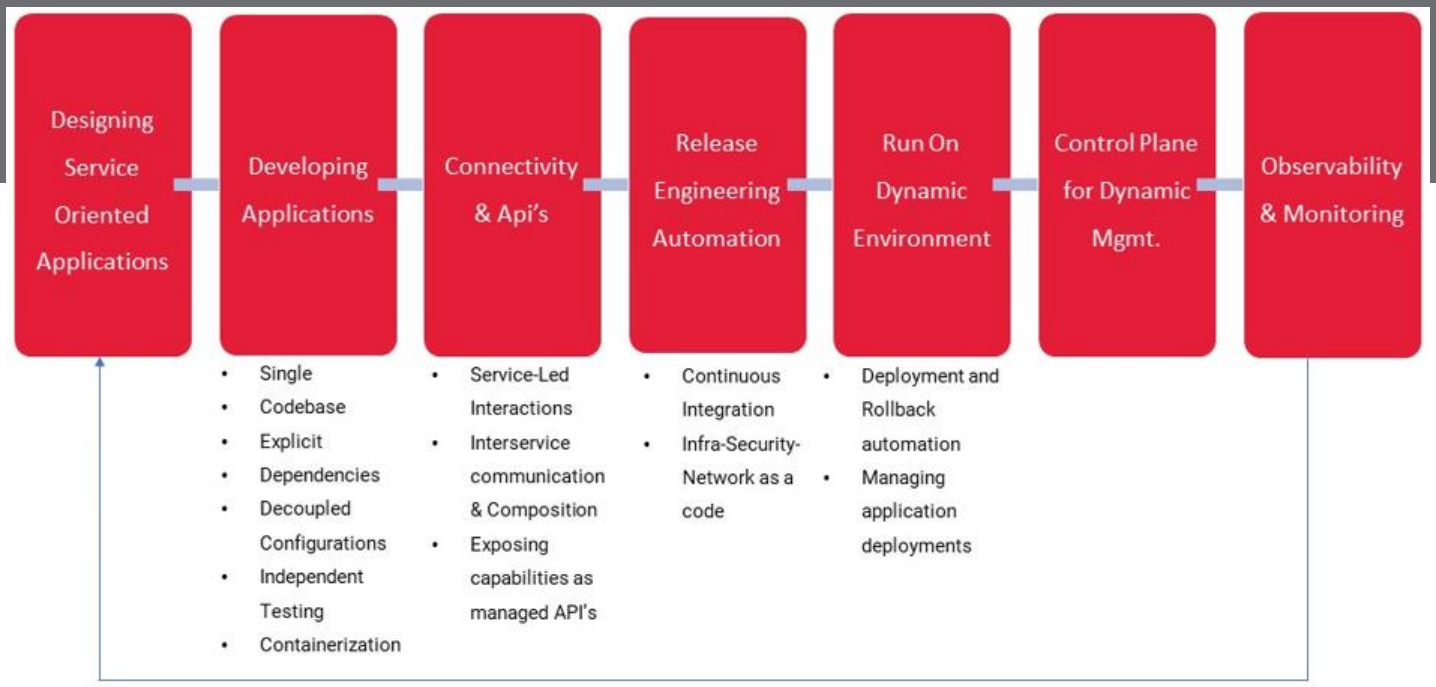
Application Transformation

The way you build, deploy, and manage applications will completely change once your teams are working in your cloud-native environment. In brief, the application layers will look like this:

APIs: The API layer provides multiple resources for a variety of applications, helping to connect data, workflows, and assets.

UX: The UX layer consists of intuitive user interfaces for dashboards, reports, customer portals, and more.

Data: The enterprise data layer stores data from all relevant systems, providing easy access for cloud native applications.



Sample flow for Cloud Native App Transformation

Challenges of Microservices Architecture

Traditionally, application development creates almost a waterfall development timeline because the development process is so tightly coupled with organization design. So, say you have four different app teams, all tightly coupled with the app's design. Even though you're shipping a single application, the process spans multiple teams across the organization.

When you're cloud native, you're running entirely on microservices architecture, but that presents some common challenges, like:

Discovery and service networking challenges, which require you to ask questions like, "Does A need to communicate with D and, if so, how will it do that?“

Data consistency, which is often the result of teams applying traditional data management techniques to cloud-native data, causing data to be redundantly stored in multiple locations. For instance, data may be stored as part of a transaction and duplicated in reporting or analytics tools.

Distributed tracing because, while distributed systems give you a stack trace for finding the cause of an error, you'll have to work backward within a microservices architecture using error messages or status codes to find problems.

Failover and recovery plan to ensure uptime. If one part of the microservices architecture fails, the development team needs to understand the dependency and have backups and restoration plans to avoid downtime.

Overhead monitoring since microservices, by nature, distribute assets and resources across a myriad of systems. There's no central control point, as there is with traditional architecture, so operational overhead must be closely managed.

While challenges like these are very real and only add to the list of traffic management and secrets management hurdles, it's important to realize that none of them are insurmountable. In fact, all of these challenges sprout from a team that isn't fully adopting CNE and is instead trying to apply traditional management and development principles to the cloud native environment.

- How do microservices pieces discover and network to one another?
  How do a partial deployment happen before going a 100% rollout?
- How to define infrastructure provisioning workflow that's hat is scalable, consistent, and reusable?
- How to move to zero trust security and manage secrets across applications & infrastructure?
- How do we get the insights to Financial operation of cloud? Make sure the quality of microservices?

Challenges of Serverless Architecture

Like microservices, serverless architecture also presents a number of challenges.

- Reduced control by nature of using a third-party vendor, which can make performance issues and other aspects more difficult to monitor and improve.
- Observability, as it's tough to monitor applications using legacy methods. Old metrics don't apply to serverless architecture, and it can be tough to use agent-based monitoring for serverless applications.
- High costs associated with vendor lock-in and "hidden" costs, like being highly dependent on API calls.

As with the challenges posed by microservices architecture, most challenges associated with serverless architecture are only an issue when teams try to apply "legacy" processes to their new cloud native environment.

Additional Considerations

As you consider the impact of application transformation during the migration to cloud-native, consider:

- API gateways and the more technical paths you can take to develop.
- The "microfrontends" framework, which uses a microapps approach, partly inspired by microservices.
- Polyglot persistence, which is the use of multiple data storage technologies based on the concept that different types of data are best handled by different storage services.

Infrastructure Transformation

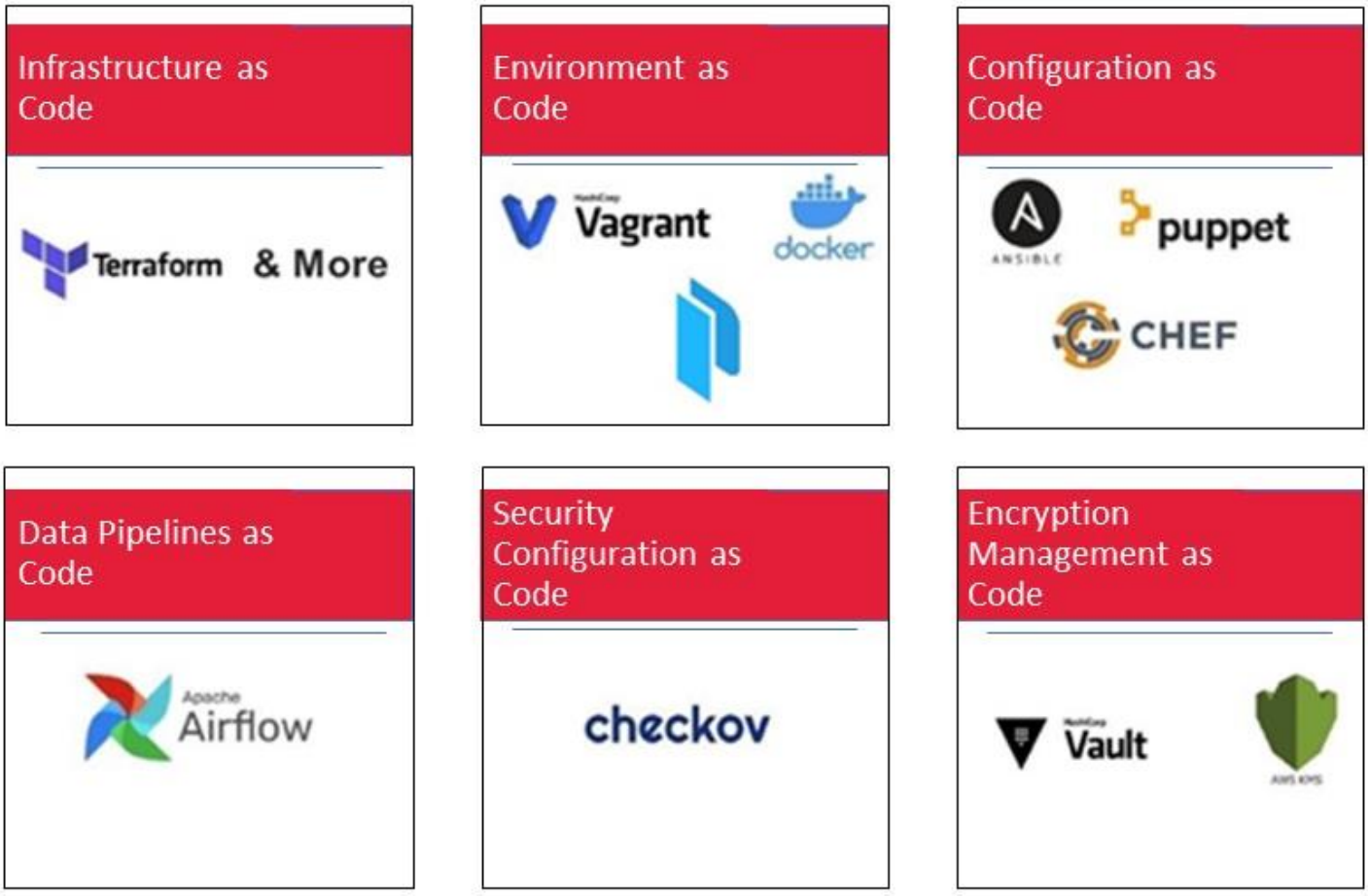
The infrastructure transformation associated with the shift to cloud native is a move from "static" to "dynamic," changing the focus from configuring and managing static resources (i.e., servers and hardware) to the process of provisioning, securing, configuration, and monitoring dynamic resources in an on-demand, as-needed manner. And driving everything as code.

Infrastructure as Code (IaC): Being cloud native means adopting the IaC model, which essentially involves applying the practices of software engineering, like versioning and testing, to your DevOps practices. Also moving to immutable infrastructure.

Immutable Security: This paradigm means embedding security into the development of an application and enforcing it continuously as your infrastructure becomes more immutable.

Service Mesh: A service mesh helps you control how various components of an application share data with each other. Unlike other approaches, a service mesh is a layer of dedicated infrastructure that's built directly into the application.

Policy as Code (PaC): Defining code to manage and automate policies. Policies can be of compliance, configurations, security, or operational excellence. By representing policies as code in text files, proven software development best practices can be adopted such as version control, automated testing, and automated deployment.



Organization Transformation

When it comes to the overall operational transformation your organization can expect when moving to a cloud-native environment, there are a handful of considerations to review. First and foremost, one thing that needs to be engrained in your systems from the start is observability.

As pointed out above, observability, distributed tracing, and data consistency -- which all tie together- are inherent challenges in a cloud-native environment posed when teams try to apply legacy processes and methods to the new architecture. As such, it's of paramount importance that you consider these challenges well in advance and work observability into the migration process from the get-go.

Assembling an SRE Team

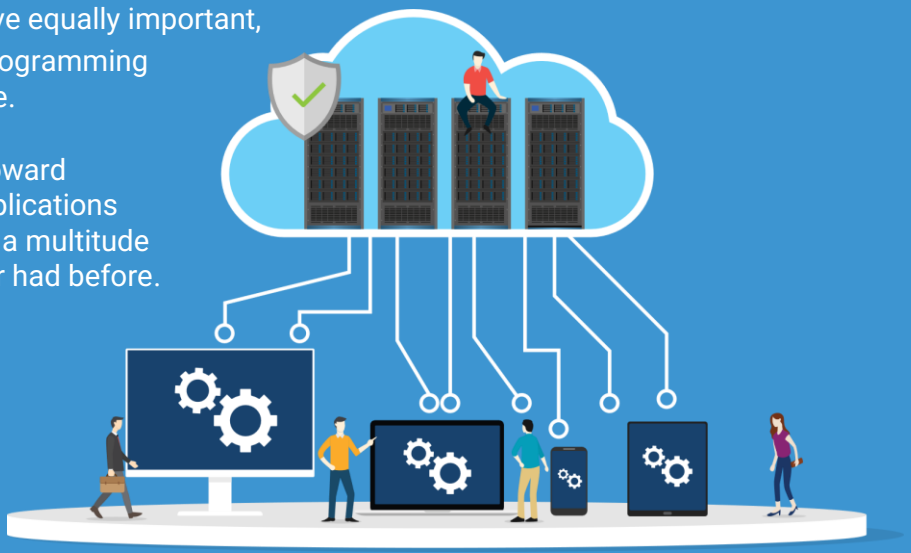
Site reliability engineering (SRE) teams apply the principles of software engineering to improve reliability by minimizing how often and how impactful failures are for applications. This isn't to be confused with a platform engineering team, which works to accelerate software delivery

The Future of Cloud Native

The cloud has already proven itself to be the future for software development. Meanwhile, IT departments everywhere recognize the cost and security benefits of moving to the cloud, while development teams realize the automation, collaboration, and coordination potential.

Looking ahead, it's further predicted that more development practices and tools will be built around native multi-cloud, allowing organizations to deploy an application across a variety of cloud platforms for public or internal use. Meanwhile, polyglot will prove equally important, enabling developers to select the ideal programming language for each component they create.

Ultimately, all cloud native trends point toward portability, scalability, and decoupling applications from the ground up to give organizations a multitude of options and flexibility like they've never had before.



Getting Results on Your CNE Journey

With so many complexities posed, from operational changes to day-to-day processes and procedures, moving to a cloud native environment requires a great deal of time and effort to plan the migration and ensure that cloud native is the right solution.

Under the umbrella of Cloud NXT.NOW we have a dedicated delivery and solution team for cloud native engineering practice. Where we have **battle-tested modernization solutions and frameworks:** 500+ projects, 1000+ cloud resources across 6+ verticals, factory frameworks, and IPs to accelerate modernizations and enhance developer and customer experiences.

With the right approach, your journey to cloud native will prove fruitful, offering flexibility, scalability, and optimization opportunities that only CNE can unlock. If you are interested to learn more, please schedule a meeting with our cloud native experts RamPrasad Nagaraja to define overall cloud native transformation operating model – organization, customer experience, data, application & infrastructure.

Or learn more

About Deep Advisory, Engineering, Platform, And Solution Offerings to Accelerate Cloud Native Transformation By 30%.

References

[i] *Understanding UCSB's federated it model. UC Santa Barbara Information Technology. (n.d.). Retrieved February 2, 2022, from https://www.it.ucsb.edu/understanding-ucsbs-federated-it-model*

[ii] *Lebeaux, R. (2014, February). DEFINITION IT centralization (information technology centralization). www.techtarget.com. Retrieved February 2, 2022, from https://www.techtarget.com/searchcio/definition/IT-centralization-information-technology-centralization*

About the author

Vineeth Rajagopal is Global Cloud Transformation Head for Tech Mahindra and has been working with technology first companies and enterprises on cloud native transformation since 2016. Prior to joining Tech Mahindra, he was the founding member and CTO of cloud native engineering focused company DigitalOnUs, Inc based in San Jose, CA.

Overall, his focus is to enable Tech Mahindra's Strategic customers the path to digital transformation, new technology adoption, build vertical cloud solutions and large-scale Hybrid cloud transformation including workload prioritization, organizational and cultural readiness, and helping enterprises to reimage business powered by cloud.

Tech Mahindra

www.youtube.com/user/techmahindra09
www.facebook.com/techmahindra
www.twitter.com/tech_mahindra
www.linkedin.com/company/tech-mahindra
www.techmahindra.com