Hello. Understanding each individual component is critical to your understanding of Apigee Edge for private cloud as a whole. In this lesson, we will focus on building that understanding. Over the next few minutes, we will be covering Apigee Edge capabilities, how each component fits into the platform, and details about the underlying technology stack. Apigee Edge has three core functional areas: API services, developer services, and analytic services. On the diagram, Apigee capabilities are shown in blue, open source capabilities are green, items in grey represent customer-specific infrastructure and API clients. Apigee capabilities such as Gateway and analytics, rely on underlying open source infrastructure to support the storage and replication of data. The Gateway sits on the critical path. Components included in this capability represent the runtime for APIs deployed on Apigee. Apigee Edge is horizontally scalable. For most scenarios, adding capacity to handle more API calls is as simple as adding additional Gateways. At some point, scaling the Gateway horizontally may require scaling the supporting infrastructure to handle the additional data being generated by the system. Components associated with other capabilities such as the UI, system management and the developer portal, only need to scale based on requirements associated with those capabilities. Horizontal scalability applies to capabilities within the region, as well as across regions. By design, Apigee Edge is active active. By that, we mean there is continuous eventually consistent data replication across data stores located in all regions. There are two regions in this diagram. API traffic is channeled to one or both regions based on customer preferences, that is usually handled by a global load balancer or DNS resolution. Regions are independent of each other. API calls and runtime execution have affinity with the region in which they arrive. Once an API call arrives at a gateway, it will stay in the region where the gateway is located. Take a moment to internalize the concepts discussed so far, as they are key to your understanding of the Apigee Edge architecture. As we discussed this diagram, think about the blue and green boxes as services or components. For now, we will focus on the components responsibilities. Later on, we will discuss the relationship as well as how to group them to form installation topologies. Let's start with the critical path. On this diagram, the critical path is represented by the horizontal line extending from the client to the backend system. The first component on the critical path is the router. The router is responsible for exposing virtual hosts and load balancing incoming requests. The message processor represents the runtime container for APIs executed on Apigee. API calls executing on the message processors may perform one or more calls to backend systems. All calls to backend systems on Apigee are originated by message processors. Some APIs generate runtime data that needs to be stored within Apigee. API policies such as API key validation, OAuth, key value map, and cache require access to a data store for the storage and retrieval of runtime data. The Apigee runtime data store is cassandra. Cassandra is part of the critical path because it is used by API policies to store data during runtime execution. The full critical path consists of the router, message processor, and cassandra services. As long as these components are up and running, others can be down or unreachable without affecting API runtime availability. API calls executed on the message processors generate analytics events. These events are asynchronously generated and consumed. The analytics pipeline is described on the bottom right part of the diagram. The first component in the analytics pipeline is apache Qpid. Qpid is a queue broker, queues are provisioned here to store analytics data. Qpid server is responsible for moving data from Qpid queues to postgreSQL. Analytics raw data on postgreSQL are eventually aggregated in batches by a service called postgres server. Aggregate data are used to power some Apigee Edge analytics reports. On the left of the diagram, we find the management components. At the center of the section is the management server. The management server exposes the management API. This API allows you to add users, deploy APIs, and perform many other actions. While performing these actions, the management server leverages cassandra, zookeeper, and openldap to read and store relevant data associated with runtime or configuration state. Zookeeper stores data related to the configuration of the system. The wiring information between components and configuration state are stored in zookeeper. Openldap stores information related to role-based access control users, roles, and permissions used for administrative access to Apigee are stored here. The Enterprise UI is the administrative UI used by API teams to develop and manage APIs. This UI consumes the management API to perform all actions. All features in the Enterprise UI are available through the management API. At the bottom of the diagram, we find the developer portal. This portal is a Drupal based application. Drupal uses postgreSQL to store data about user accounts, sessions, and modules. Take some time to review the component diagram in detail, your understanding of how Apigee Edge components function and relate to each other is key to your understanding and management of the platform. For more information on this topic, refer to our documentation. If you have any questions, please post them on our community. Thanks for watching.