This chapter covers architectural patterns that describe how cloud applications as described in Chap. 4, can be managed automatically by separate components (Fig. 5.1). These management components (Sect. 5.2) handle the automated management of cloud-native applications regarding dynamic elasticity, resiliency, updates etc. Due to the pay-per-use property of cloud applications covered in Sect. 1.1, scaling tasks should be automated, because the number of provisioned IT resources, i.e., the number of provisioned virtual servers, the size of booked storage or the number of application component instances directly affects the runtime costs of an application. Furthermore, environment-based availability (98) assurances, where individual cloud resources can fail at any time, or a node-based availability (95) that...
does not meet requirements of an application, as well as network partitions, may create the need to monitor applications and automatically react to resource failures.

The management processes (Sect. 5.3) executed by management components are described as separate patterns. They address tasks such as elasticity management, version updates, or failure resiliency. In the overview, we introduce the covered patterns and show how the management components are integrated into a cloud native application.

5.1 Overview of Application Management Patterns

Management components (Sect. 5.2) form the execution environment for management processes (Sect. 5.3). Therefore, they describe the architectural components that enable the automated execution of management processes handling application components and system resources. Provider adapters (243) are used to encapsulate provider interfaces. The encapsulated functionality may then be accessed from other non-provider-specific management components. Especially, encapsulation can be used to trigger other management processes when a certain provider function is called, for example, if a failure in a provider-supplied function has to be addressed. Most management patterns handle a large set of application components and, often, a coordinated behavior of these components is required. Managed configurations (247) centrally control the behavior of application component instances and distributed management components in a unified fashion. An elasticity manager (250) enables automated horizontal scalability based on resource utilization. To manage elasticity and distribute workload across horizontally scaled instances based on the number of handled requests the elastic load balancer (254) is needed. When communicating asynchronously via queues, similar behavior can be realized via an elastic queue (257) that dynamically adjusts the number of message processing components depending on the number of pending requests. If one of these elasticity management components is used, it can implement the elasticity management process (267) describing the basic process to be followed when scaling and application up or down. This process can then be extended further. A standby pooling process (279) optimizes the provisioning and decommissioning of resources by considering the timeframe for which resources have been paid for prior to decommissioning them. Additionally, it can be used to keep component instances on standby to speed up provisioning for critical application components. If new resources still cannot be obtained fast enough from the cloud provider during a workload increase, the feature flag management process (271) enables an application to degrade gracefully by replacing or disabling less important functionality. Finally, when new application component versions have been developed, an update transition process (275) can be used to seamlessly switch between different versions with minimal or no downtime of the application component.

Similar to the elasticity considerations, availability of applications is typically a very important topic. Depending on the requirements of an application or platform, the node-based availability (95) assured by elastic infrastructures (87) or elastic