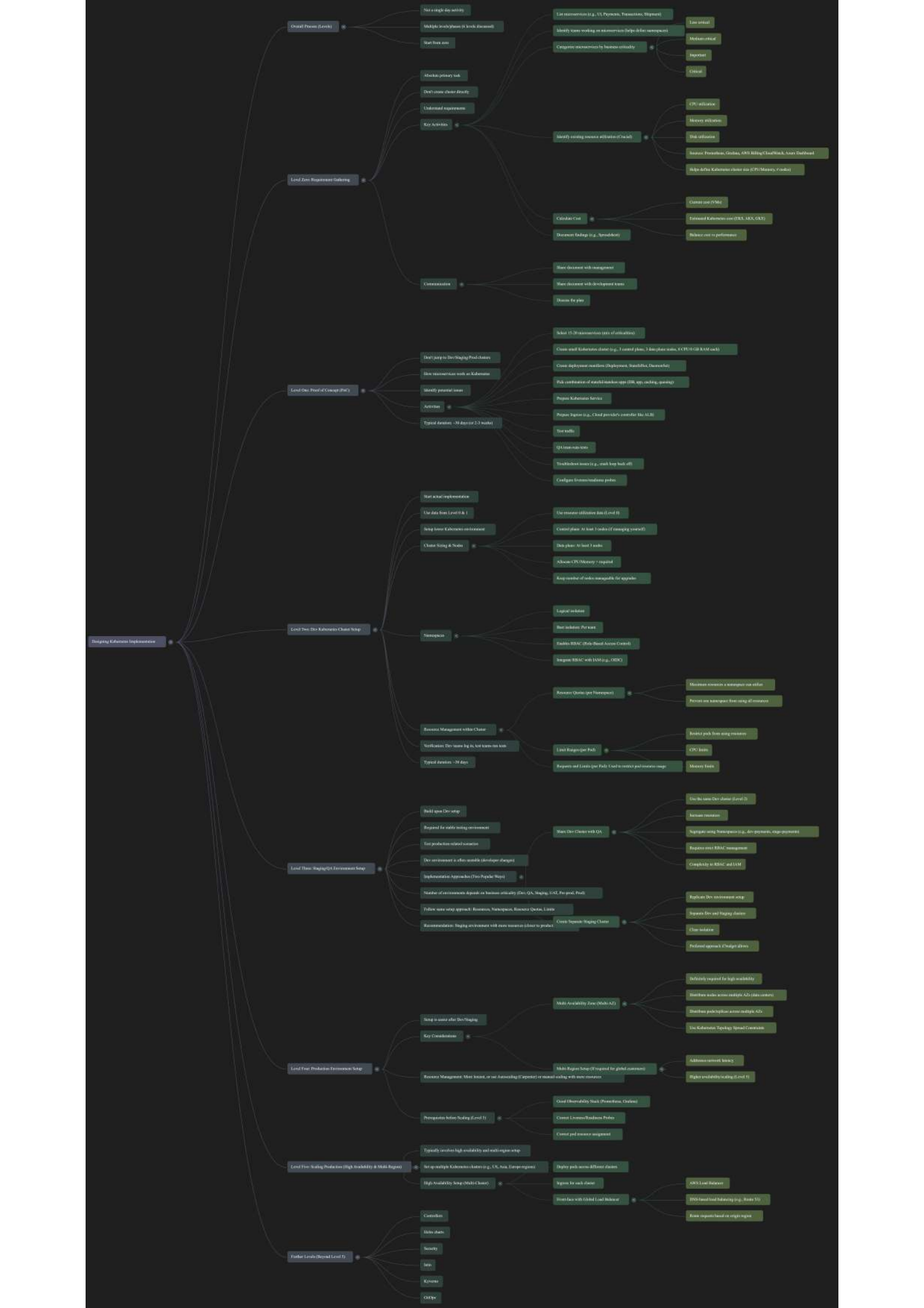
Designing Kubernetes Implementation
Overall Process (Levels)
Not a single day activity
Multiple levels/phases (6 levels discussed)
Start from zero
Level Zero: Requirement Gathering
Absolute primary task
Don't create cluster directly
Understand requirements
Key Activities
List microservices (e.g., UI, Payments, Transactions, Shipment)
Identify teams working on microservices (helps define namespaces)
Categorize microservices by business criticality
Low critical
Medium critical
Important
Critical
Identify existing resource utilization (Crucial)
CPU utilization
Memory utilization
Disk utilization
Sources: Prometheus, Grafana, AWS Billing/CloudWatch, Azure Dashboard
Helps define Kubernetes cluster size (CPU/Memory, # nodes)
Calculate Cost
Current cost (VMs)
Estimated Kubernetes cost (EKS, AKS, GKE)
Balance cost vs performance
Document findings (e.g., Spreadsheet)
Communication
Share document with management
Share document with development teams
Discuss the plan
Level One: Proof of Concept (PoC)
Don't jump to Dev/Staging/Prod clusters
Slow microservices work on Kubernetes
Identify potential issues
Activities
Select 15-20 microservices (mix of criticalities)
Create small Kubernetes cluster (e.g., 1 control plane, 2 data plane nodes, 8 CPU/16 GB RAM each)
Create deployment manifests (Deployment, StatefulSet, DaemonSet)
Pick combination of stateful/stateless apps (DB, app, caching, queuing)
Prepare Kubernetes Service
Prepare Ingress (e.g., Cloud provider's controller like ALB)
Test traffic
QA team runs tests
Troubleshoot issues (e.g., crash loop back off)
Configure liveness/readiness probes
Typical duration: ~30 days (or 2-3 weeks)
Level Two: Dev Kubernetes Cluster Setup
Start actual implementation
Use data from Level 0 & 1
Setup lower Kubernetes environment
Cluster Sizing & Scales
Use resource utilization data (Level 0)
Control plane: At least 3 nodes (if managing yourself)
Data plane: At least 3 nodes
Allocate CPU/Memory > required
Keep number of nodes manageable for upgrades
Namespaces
Logical isolation
Blast isolation: Per team
Enables RBAC (Role-Based Access Control)
Integrate RBAC with IAM (e.g., OIDC)
Resource Management within Cluster
Resource Quotas (per Namespace)
Maximum resources a namespace can utilize
Prevents one namespace from using all resources
Verification: Dev teams log in, test teams run tests
Limit Ranges (per Pod)
Restrict pods from using resources
CPU limits
Typical duration: ~30 days
Requests and Limits (per Pod): Used to restrict pod resource usage
Memory limits
Level Three: Staging/QA Environment Setup
Build upon Dev setup
Required for stable testing environment
Test production related scenarios
Dev environment is often unstable (developer changes)
Implementation Approaches (Two Popular Ways)
Share Dev Cluster with QA
Use the same Dev cluster (Level 2)
Increase resources
Segregate using Namespaces (e.g., dev-payments, stage-payments)
Requires strict RBAC management
Complexity in RBAC and IAM
Create Separate Staging Cluster
Replicate Dev environment setup
Separate Dev and Staging clusters
Clear isolation
Preferred approach if budget allows
Number of environments depends on business criticality (Dev, QA, Staging, UAT, Pre-prod, Prod)
Follow same setup approach: Resources, Namespaces, Resource Quotas, Limits
Recommendation: Staging environment with more resources (closer to prod)
Level Four: Production Environment Setup
Setup is easier after Dev/Staging
Key Considerations
Multi-Availability Zone (Multi-AZ)
Definitely required for high availability
Distribute nodes across multiple AZs (data centers)
Distribute pods/replicas across multiple AZs
Use Kubernetes Topology Spread Constraints
Multi-Region Setup (If required for global customers)
Addresses network latency
Higher availability/scaling (Level 5)
Resource Management: More honest, or use Autoscaling (Carpenter) or manual scaling with more resources
Prerequisites before Scaling (Level 5)
Good Observability Stack (Prometheus, Grafana)
Correct Liveness/Readiness Probes
Correct pod resource assignment
Level Five: Scaling Production (High Availability & Multi-Region)
Typically involves high availability and multi-region setup
Set up multiple Kubernetes clusters (e.g., US, Asia, Europe regions)
Deploy pods across different clusters
High Availability Setup (Multi-Cluster)
Ingress for each cluster
Front face with Global Load Balancer
AWS Load Balancer
DNS-based load balancing (e.g., Route 53)
Route requests based on origin region
Further Levels (Beyond Level 5)
Controllers
Helm charts
Security
Istio
Kyverno
OPA/gatekeeper