User Stories : ClusterMaster SaaS Hub

Initial Project Setup and Infrastructure

This epic covers foundational setup tasks including repository initialization, development environment setup, CI/CD pipeline configuration, database migration framework, testing frameworks, code quality tools, and documentation setup to ensure a robust project foundation.

7 stories

Spike: Documentation Framework Setup

medium

As a technical writer, I want to set up documentation frameworks for API docs and user guides, so that project documentation is accessible and maintainable.

Acceptance Criteria:

- ✓ API docs auto-generated and accessible via UI
- ✓ User guides cover initial features
- ✓ Documentation is version controlled and updated

Story Points: 3

Spike: Code Quality Tools Setup

medium

As a developer, I want to integrate ESLint, Prettier, and other code quality tools, so that code style and quality are maintained consistently.

Acceptance Criteria:

- ✓ Code style violations are detected and reported
- ✓ Developers are prevented from committing bad style
- ✓ Documentation on coding standards is available

Story Points: 2

Spike: Testing Framework Setup

high

As a QA engineer, I need to set up unit, integration, and E2E testing frameworks, so that code quality and functionality are verified automatically.

Acceptance Criteria:

- ✓ Tests run automatically in CI pipeline
- ✓ Code coverage reports generated
- ✓ Test failures cause pipeline to fail

Story Points: 5

Spike: Database Migration Framework Setup

high

As a backend engineer, I need to set up a database migration framework, so that schema changes are version-controlled and deployable.

Acceptance Criteria:

- ✓ Migrations can be applied and rolled back reliably
- ✓ CI pipeline runs migrations on test databases
- ✓ Migration scripts stored in version control

Story Points: 3

Spike: CI/CD Pipeline Setup

high

As a DevOps engineer, I need to configure CI/CD pipelines using GitHub Actions and ArgoCD, so that code is automatically built, tested, and deployed.

Acceptance Criteria:

- ✓ Pull requests trigger CI workflows with status checks
- ✓ Successful builds trigger deployment to dev environment
- ✓ Pipeline includes security scanning and code quality checks

Story Points: 5

Spike: Development Environment Setup

high

As a developer, I need to set up local development environments with Docker and necessary tools, so that development is consistent and efficient.

Acceptance Criteria:

- ✓ Developers can run frontend and backend locally with one command
- ✓ Environment variables and secrets management documented
- ✓ Docker images are optimized for local development

Story Points: 3

Spike: Repository Initialization and Project Scaffolding

high

As a developer, I need to set up the initial project repository and scaffolding, so that the team has a standardized codebase to start development.

Acceptance Criteria:

- ✓ Repository is created with README and license
- ✓ Basic React frontend and Node.js backend scaffolds are committed
- ✓ Branch protection rules are configured

Story Points: 3

Reliability and Availability

This epic ensures the platform is reliable and highly available, minimizing downtime and ensuring continuous Kubernetes operations management through redundancy, failover, error handling, and monitoring.

3 stories

Implement Continuous Monitoring of Platform Health

high

As the system, I want continuous monitoring of platform health with alerts, so that issues are detected and resolved proactively.

Dependencies:

infra-1-...

Acceptance Criteria:

- Functional: Platform health metrics (CPU, memory, latency, error rates) are collected and visualized.
- Functional: Alerts are configured for critical health degradations.
- Technical: Prometheus and Grafana are deployed for monitoring and dashboards.
- Technical: Alertmanager integrates with NotificationService for alert delivery.
- Technical: Monitoring covers all microservices, databases, and infrastructure.

Story Points: 8

Estimated Effort: 8 hours

Implement Robust Error Handling and Recovery

high

As the system, I want robust error handling and automated recovery for failures, so that operational disruptions are minimized.

Dependencies:

e1e5e730... b2fe9d34...

Acceptance Criteria:

- Functional: Services handle errors gracefully with retries and circuit breakers.
- Functional: Failed workflows trigger rollback or compensation actions.
- Technical: Implement circuit breaker pattern in ClusterConnectorManager and API Gateway.
- Technical: Workflow Engine supports retry policies and failure notifications.
- Technical: Errors are logged with context and severity.

Story Points: 8

Estimated Effort: 8 hours

Implement Redundancy and Failover Mechanisms

high

As the system, I want core services and databases to have redundancy and failover, so that platform availability meets SLA requirements.

Dependencies:

infra-1-...

Acceptance Criteria:

- ✓ Functional: Services are deployed in multi-AZ clusters with failover capabilities.
- ✓ Functional: Databases support streaming replication and automatic failover.
- ✓ Technical: Kubernetes deployments use multi-AZ node pools with pod anti-affinity.
- ✓ Technical: Health checks trigger failover and self-healing actions.
- ✓ Technical: Failover events are logged and alerted.

Story Points: 8

Estimated Effort: 8 hours

Security and Compliance

This epic ensures secure access, data protection, and compliance enforcement across the platform and managed clusters, including authentication, authorization, encryption, audit logging, and regulatory compliance.

3 stories

Implement Audit Logging and Compliance Reporting

high

As a compliance officer, I want immutable audit logs of all critical actions and compliance reports, so that regulatory requirements are met.

Dependencies:

0b9de92e...

Acceptance Criteria:

- ✓ Functional: All user actions, policy changes, and system events are logged immutably.
- ✓ Functional: Audit logs are queryable and exportable for compliance audits.
- ✓ Technical: Logs are stored securely with encryption at rest.
- ✓ Technical: Compliance reports are generated per GDPR, HIPAA, ISO 27001, and SOC 2 standards.
- ✓ Technical: Audit logging integrates with PolicyManager and Governance services.

Story Points: 8

Estimated Effort: 8 hours

Secure Authentication and Authorization with OAuth2 and SAML

high

As a user, I want to authenticate securely using OAuth2 or SAML SSO, so that my access is protected and compliant with enterprise policies.

Acceptance Criteria:

- ✓ Functional: Users can log in via OAuth2 or SAML SSO providers.
- ✓ Functional: Multi-factor authentication (MFA) is supported and enforced.
- ✓ Technical: AuthService issues JWT tokens with appropriate claims and expiry.
- ✓ Technical: Tokens are validated at API Gateway and services.
- ✓ Technical: Authentication failures are logged and monitored.

Story Points: 8

Estimated Effort: 8 hours

Implement Role-Based Access Control (RBAC)

high

As a security administrator, I want RBAC enforced across all APIs and UI components, so that users have least privilege access.

Dependencies:

infra-1-...

Acceptance Criteria:

- ✓ Functional: Users have roles assigned with specific permissions.
- ✓ Functional: Access to API endpoints and UI features is restricted based on roles.
- ✓ Technical: RBAC policies are stored in PostgreSQL and enforced in API Gateway and services.
- ✓ Technical: Unauthorized access attempts are logged and alerted.
- ✓ Technical: RBAC integrates with OAuth2 and SAML authentication flows.

Story Points: 8

Estimated Effort: 8 hours

Performance and Scalability

This epic addresses non-functional requirements to ensure the platform performs efficiently and scales to support mid-to-large enterprises with thousands of clusters and concurrent users, maintaining responsive UI and backend performance.

2 stories

Support Horizontal and Vertical Scaling of Services

high

As the system, I want backend services to scale horizontally and vertically to handle increasing load, so that platform remains stable and performant.

Dependencies:

infra-1-...

Acceptance Criteria:

- ✓ Functional: Services scale automatically based on CPU, memory, and custom metrics.
- ✓ Technical: Kubernetes HPA configured for microservices with appropriate thresholds.
- ✓ Technical: Database clusters support replication and scaling.
- ✓ Technical: Observability pipelines scale with workload.
- ✓ Technical: Auto-scaling tested under simulated load.

Story Points: 8

Estimated Effort: 8 hours

Ensure Dashboard Loads Within 2 Seconds Under Load

high

As the system, I want the dashboard to load in under 2 seconds even with 1000+ clusters, so that users have a responsive experience.

Dependencies:

3b6551c9... 4585ec35...

Acceptance Criteria:

- ✓ Functional: Dashboard UI loads and renders key data within 2 seconds for typical user sessions.
- ✓ Technical: Backend APIs respond within 500ms for 95% of requests under load.
- ✓ Technical: Caching strategies (Redis) are implemented for hot data.
- ✓ Technical: Load testing simulates 10,000 concurrent users and 1,000 clusters.
- ✓ Technical: Performance bottlenecks are identified and mitigated.

Story Points: 8

Estimated Effort: 8 hours

Hybrid and Multi-Cloud Environment Support

This epic ensures the platform supports managing Kubernetes clusters deployed on-premise and across multiple cloud providers including AWS, GCP, and Azure, providing seamless and consistent operations across heterogeneous environments.

3 stories

Ensure Consistent Operations Across Heterogeneous Environments

high

As a platform engineer, I want consistent lifecycle management and observability features across all clusters, so that operational processes are unified.

Dependencies:

b2fe9d34... 8a04b05f...

Acceptance Criteria:

- ✓ Functional: Lifecycle operations (deploy, scale, upgrade) behave consistently across all cluster types.
- ✓ Functional: Observability data is normalized and aggregated from all environments.
- ✓ Technical: Cluster abstraction layer provides unified API for services.
- ✓ Technical: Tests validate consistent behavior across on-prem and cloud clusters.
- ✓ Technical: UI shows unified views without environment-specific discrepancies.

Story Points: 8

Estimated Effort: 8 hours

Handle Provider-Specific APIs and Authentication

high

As the system, I want to abstract provider-specific Kubernetes APIs and authentication mechanisms, so that operations are consistent regardless of cluster location.

Dependencies:

b2fe9d34...

Acceptance Criteria:

- ✓ Functional: Platform translates generic cluster operations to provider-specific API calls.
- ✓ Functional: Authentication tokens and credentials are securely managed and rotated.
- ✓ Technical: ProviderIntegrationAdapter abstracts AWS EKS, GCP GKE, Azure AKS, and on-prem APIs.
- ✓ Technical: Implements circuit breaker pattern for external API calls.
- ✓ Technical: Logs and alerts on authentication failures or API errors.

Story Points: 8

Estimated Effort: 8 hours

Connect and Manage Clusters Across On-Premise and Cloud Environments

high

As a platform engineer, I want to register and manage Kubernetes clusters from on-premise and multiple cloud providers, so that I have unified control over my infrastructure.

Acceptance Criteria:

- ✓ Functional: Users can register clusters with provider-specific credentials and metadata.
- ✓ Functional: Platform displays clusters from all environments in unified dashboard.
- ✓ Technical: ClusterConnectorManager handles secure connectivity and API proxying.
- ✓ Technical: Supports authentication methods including service accounts, IAM roles, and kubeconfigs.
- ✓ Technical: Cluster metadata stored in PostgreSQL with provider and region details.

Story Points: 13

Estimated Effort: 13 hours

Application Deployment and Upgrade Workflow

This epic covers the guided process for selecting applications, configuring deployment or upgrade settings, and initiating automated orchestration across clusters with integrated health monitoring and scaling support.

4 stories

Integrate Health Monitoring During Deployment and Upgrade

high

As an SRE, I want health monitoring integrated into deployment and upgrade workflows, so that I can ensure application stability during changes.

Dependencies:

e1e5e730... 5d41ad59...

Acceptance Criteria:

- ✓ Functional: Health checks are performed at each step of deployment and upgrade.
- ✓ Functional: Workflow pauses or rolls back if health degrades beyond thresholds.
- ✓ Technical: HealthMonitor service provides real-time status to Workflow Engine.
- ✓ Technical: Alerts are generated for health issues during workflows.
- ✓ Technical: Frontend displays health status and alerts during operations.

Story Points: 8

Estimated Effort: 8 hours

Automate Orchestration of Deployment and Upgrade Processes

high

As the system, I want to orchestrate deployment and upgrade workflows automatically, so that operations are consistent, reliable, and scalable.

Dependencies:

a4889d51...

Acceptance Criteria:

- ✓ Functional: Deployment and upgrade workflows execute steps automatically with status updates.
- ✓ Functional: Supports rollback on failure and health check integration.
- ✓ Technical: Uses Workflow Engine (Temporal) to manage multi-step orchestration.
- ✓ Technical: Integrates with ClusterConnectorManager for cluster API calls.
- ✓ Technical: Deployment events are logged and accessible via API.

Story Points: 13

Estimated Effort: 13 hours

Provide Configuration Options for Deployment and Upgrades

high

As a platform engineer, I want to configure deployment and upgrade settings such as replicas and environment variables, so that I can customize application behavior per cluster.

Dependencies:

81435c36...

Acceptance Criteria:

- ✓ Functional: UI allows editing configuration overrides before deployment or upgrade.
- ✓ Functional: Configuration schema validation is performed client and server side.
- ✓ Technical: AppConfigManager validates and stores configuration JSON schemas.
- ✓ Technical: API endpoints accept and validate config overrides in deployment requests.
- ✓ Technical: Configuration changes are versioned and auditable.

Story Points: 8

Estimated Effort: 8 hours

Implement Application Selection Interface

high

As a platform engineer, I want to browse and select applications for deployment or upgrade, so that I can manage stateful workloads efficiently.

Acceptance Criteria:

- ✓ Functional: UI lists available applications with search and filtering capabilities.
- ✓ Functional: Application details and versions are viewable before selection.
- ✓ Technical: Frontend fetches application data from /api/v1/applications and /api/v1/applications/:id/versions.
- ✓ Technical: API responses are paginated and cached for performance.
- ✓ Technical: RBAC enforced on application data access.

Story Points: 5

Estimated Effort: 5 hours

Intuitive and Illustrative User Interface

This epic focuses on delivering an easy-to-use interface with visual workflows, step-by-step guidance, and clear visual cues that simplify complex Kubernetes operations, improving productivity and reducing the learning curve for platform teams and SREs.

3 stories

Design Clear Visual Cues and Illustrative Elements

high

As a platform engineer, I want clear visual cues and illustrative elements in the UI, so that I can understand system states and actions intuitively.

Acceptance Criteria:

- ✓ Functional: UI uses icons, colors, and animations to indicate statuses and actions.
- ✓ Functional: Visual elements are consistent across all modules and responsive.
- ✓ Technical: UIComponentLibrary provides reusable components for visual cues.
- ✓ Technical: Design follows UX best practices and is tested with user feedback.
- ✓ Technical: Components are optimized for performance and accessibility.

Story Points: 8

Estimated Effort: 8 hours

Provide Step-by-Step Guidance and Onboarding Tours

high

As a new user, I want step-by-step guidance and onboarding tours, so that I can learn to use the platform quickly and effectively.

Acceptance Criteria:

- ✓ Functional: Onboarding tours guide users through key features and workflows.
- ✓ Functional: Users can track progress and resume tours at any time.
- ✓ Technical: TourProgressTracker stores user progress in backend and syncs with UI.
- ✓ Technical: Tours are configurable and extensible for future features.
- ✓ Technical: UI supports keyboard navigation and screen readers.

Story Points: 8

Estimated Effort: 8 hours

Implement Visual Workflows for Key Operations

high

As a platform engineer, I want visual workflows guiding me through complex operations like deploying applications and managing clusters, so that I can complete tasks efficiently and with confidence.

Dependencies:

epic-2-u... epic-6-u...

Acceptance Criteria:

- ✓ Functional: Visual workflows are available for deployment, scaling, upgrading, and governance tasks.
- ✓ Functional: Workflows provide step-by-step guidance with progress indicators.
- ✓ Technical: UIWorkflowGuide component manages workflow states and transitions.
- ✓ Technical: Workflows are responsive and accessible per WCAG 2.1 AA standards.
- ✓ Technical: Frontend uses React with interactive UI components and animations.

Story Points: 13

Estimated Effort: 13 hours

Centralized Governance and Policy Management

This epic covers centralized policy management to enforce consistent governance and compliance across hybrid and multi-cloud Kubernetes environments, including policy definition, enforcement, audit logging, and reporting.

3 stories

Provide Audit Trails and Compliance Reporting

high

As a compliance officer, I want to view audit trails and generate compliance reports, so that I can demonstrate adherence to policies and regulatory requirements.

Dependencies:

8452d97a... bbe13fa7...

Acceptance Criteria:

- Functional: Audit logs capture all policy changes, enforcement actions, and user activities.
- Functional: Users can query audit logs with filters by user, action, resource, and time.
- Functional: Compliance reports can be generated and exported in PDF/CSV formats.
- Technical: Audit logs are immutable and stored securely in PostgreSQL.
- Technical: Reporting APIs provide aggregated compliance data with pagination.

Story Points: 8

Estimated Effort: 8 hours

Enforce Compliance Across All Managed Clusters

high

As the system, I want to enforce governance policies consistently across all managed clusters, so that compliance is maintained in hybrid and multi-cloud environments.

Dependencies:

8452d97a...

Acceptance Criteria:

- Functional: Policies are pushed and enforced on clusters using policy engines (OPA/Kyverno).
- Functional: Enforcement status is reported back to the platform for audit and alerting.
- Technical: PolicyEnforcer integrates with ClusterConnectorManager to apply policies.
- Technical: Enforcement failures trigger alerts and are logged in audit logs.
- Technical: Supports multi-cloud provider API differences transparently.

Story Points: 13

Estimated Effort: 13 hours

Define and Manage Governance Policies Centrally

high

As a compliance officer, I want to create and manage governance policies centrally, so that I can enforce consistent rules across all clusters.

Acceptance Criteria:

- ✓ Functional: Users can create, edit, and delete governance policies via UI and API.
- ✓ Functional: Policies support types such as security, compliance, resource limits, and custom rules.
- ✓ Technical: PolicyManager service stores policies in PostgreSQL with JSONB definitions.
- ✓ Technical: API endpoints support CRUD operations with RBAC enforcement.
- ✓ Technical: Input validation and sanitization are applied to policy definitions.

Story Points: 8

Estimated Effort: 8 hours

Observability Tools

This epic includes features for aggregating metrics, logs, and traces from hybrid and multi-cloud Kubernetes clusters, providing monitoring dashboards, troubleshooting tools, and alerting capabilities to enhance visibility and operational efficiency.

3 stories

Enable Alerting Based on Observability Data

high

As an SRE, I want to configure alerts based on metrics, logs, and traces, so that I am proactively notified of issues in stateful workloads.

Dependencies:

8a04b05f...

Acceptance Criteria:

- ✓ Functional: Users can define alert rules with thresholds and conditions.
- ✓ Functional: Alerts trigger notifications via in-app, email, or webhook channels.
- ✓ Technical: AlertRuleEngine evaluates observability data streams and generates alerts.
- ✓ Technical: NotificationService integrates with Slack, Teams, and SMTP for alert delivery.
- ✓ Technical: Alert lifecycle (active, resolved) is tracked and displayed in UI.

Story Points: 8

Estimated Effort: 8 hours

Provide Monitoring Dashboards and Troubleshooting Tools

high

As an SRE, I want dashboards and tools to monitor metrics, logs, and traces, so that I can troubleshoot and optimize stateful workloads effectively.

Dependencies:

8a04b05f...

Acceptance Criteria:

- Functional: Pre-built dashboards display cluster and application metrics, logs, and traces.
- Functional: Users can filter and search observability data by cluster, application, and time range.
- Technical: Frontend integrates Grafana dashboards and custom React components.
- Technical: APIs support querying observability data with pagination and filtering.
- Technical: Dashboard load time is under 2 seconds for typical queries.

Story Points: 13

Estimated Effort: 13 hours

Collect and Unify Metrics, Logs, and Traces

high

As the system, I want to collect and unify observability data from all managed clusters, so that users have a holistic view of workload performance.

Dependencies:

infra-1-...

Acceptance Criteria:

- Functional: Metrics, logs, and traces are collected from on-prem and cloud clusters.
- Functional: Data is normalized and stored in appropriate backends (TimescaleDB, MinIO, etc.).
- Technical: ObservabilityCollector integrates with Prometheus, Loki, and Jaeger APIs.
- Technical: Data pipelines use Kafka event bus for reliable ingestion.
- Technical: Data retention policies are configurable per tenant.

Story Points: 13

Estimated Effort: 13 hours

Simplified Cluster Lifecycle Management

This epic encompasses streamlined processes for deploying, scaling, and upgrading stateful applications across hybrid and multi-cloud Kubernetes environments. It automates orchestration, ensures proper scaling, and integrates health monitoring to reduce manual effort and errors.

4 stories

Provide Health Monitoring During Lifecycle Operations

high

As an SRE, I want continuous health monitoring during deployments, scaling, and upgrades, so that I can detect and respond to issues promptly.

Dependencies:

ff1f1719... c05df21e...

Acceptance Criteria:

- ✓ Functional: Health status of applications is monitored and displayed during lifecycle operations.
- ✓ Functional: Alerts are generated if health degrades during operations.
- ✓ Technical: HealthMonitor service integrates with ObservabilityService to fetch real-time metrics.
- ✓ Technical: Health data is pushed to Workflow Engine to influence orchestration decisions.
- ✓ Technical: Frontend UI shows health indicators and alerts during operations.

Story Points: 8

Estimated Effort: 8 hours

Automate Upgrade Workflow for Stateful Applications

high

As a platform engineer, I want to upgrade stateful applications with automated orchestration and rollback support, so that upgrades are safe and minimize downtime.

Dependencies:

ff1f1719...

Acceptance Criteria:

- ✓ Functional: User can initiate upgrades selecting target version and configuration.
- ✓ Functional: Upgrade process is orchestrated with health monitoring and rollback on failure.
- ✓ Technical: Workflow Engine manages upgrade steps with integration to AppDeploymentService.
- ✓ Technical: Supports hybrid and multi-cloud environments with provider-specific API handling.
- ✓ Technical: Deployment events and logs are recorded and accessible.

Story Points: 13

Estimated Effort: 13 hours

Enable Scaling Operations for Stateful Applications

high

As an SRE, I want to scale stateful applications up or down across clusters, so that resource usage matches workload demands.

Dependencies:

ff1f1719...

Acceptance Criteria:

- ✓ Functional: User can initiate scaling operations with desired replica counts.
- ✓ Functional: Scaling actions are automated and reflected in cluster state.
- ✓ Technical: Scaling commands are executed via ClusterLifecycleController and ClusterConnectorManager.
- ✓ Technical: Health monitoring integration ensures scaling does not degrade application availability.
- ✓ Technical: Rollback mechanisms are in place for failed scaling operations.

Story Points: 8

Estimated Effort: 8 hours

Support Stateful Application Deployment Workflow

high

As a platform engineer, I want to deploy stateful applications across clusters using a guided workflow, so that deployments are reliable and consistent across hybrid environments.

Dependencies:

epic-6-u... epic-1-u...

Acceptance Criteria:

- ✓ Functional: User can select an application and initiate deployment with configuration options.
- ✓ Functional: Deployment process is automated and orchestrated across selected clusters.
- ✓ Technical: Uses AppDeploymentService and Workflow Engine to manage deployment steps.
- ✓ Technical: Supports multi-cloud API compatibility for AWS, GCP, Azure, and on-prem clusters.
- ✓ Technical: Deployment status and logs are accessible via API and UI.

Story Points: 13

Estimated Effort: 13 hours

Cluster Dashboard

This epic covers all features related to the Cluster Dashboard, providing platform teams and SREs with a clear, real-time overview of all managed Kubernetes clusters, their health status, key performance metrics, and critical alerts. It enables quick assessment and situational awareness of the cluster landscape.

4 stories

Provide Snapshot Overview for Quick Cluster Assessment

high

As a platform engineer, I want a snapshot summary on the dashboard showing counts of clusters by health status and number of critical alerts, so that I can quickly understand the overall cluster landscape.

Dependencies:

3b6551c9... c05df21e...

Acceptance Criteria:

- ✓ Functional: Dashboard includes summary cards showing total clusters, healthy clusters, clusters with warnings, and critical alerts count.
- ✓ Functional: Summary updates dynamically with real-time data.
- ✓ Technical: Summary data is aggregated efficiently from cluster and alert databases.
- ✓ Technical: API endpoint /api/v1/dashboard/summary provides aggregated data with response time < 200ms.

- ✓ Technical: Frontend renders summary cards with responsive design.

Story Points: 5

Estimated Effort: 5 hours

Display Critical Alerts and Notifications on Dashboard

high

As an SRE, I want to see critical alerts and notifications related to clusters on the dashboard, so that I can quickly identify and respond to issues.

Dependencies:

3b6551c9...

Acceptance Criteria:

- ✓ Functional: Dashboard shows active critical alerts with severity and message.
- ✓ Functional: Alerts update in real-time as they are generated or resolved.
- ✓ Technical: Alerts are aggregated from ClusterAlertService and Observability AlertRuleEngine.
- ✓ Technical: Alert data is stored in PostgreSQL and pushed via event-driven Kafka streams.
- ✓ Technical: Frontend displays alerts with visual indicators and allows filtering by severity.

Story Points: 8

Estimated Effort: 8 hours

Show Key Performance Metrics per Cluster

high

As a platform engineer, I want to view key performance metrics (CPU usage, memory usage, network IO) for each cluster on the dashboard, so that I can monitor resource utilization at a glance.

Dependencies:

3b6551c9...

Acceptance Criteria:

- ✓ Functional: Dashboard displays CPU, memory, and network IO metrics for each cluster.
- ✓ Functional: Metrics are updated at least every minute.
- ✓ Technical: Metrics are aggregated from TimescaleDB and Prometheus via Observability Service APIs.
- ✓ Technical: Frontend uses charting libraries (e.g., Chart.js) to visualize metrics responsively.
- ✓ Technical: API response time for metrics data is under 500ms for 95% of requests.

Story Points: 8

Estimated Effort: 8 hours

Display All Managed Clusters with Health Status

high

As a platform engineer, I want to see a list of all managed Kubernetes clusters with their current health status, so that I can quickly assess the operational state of my cluster landscape.

Acceptance Criteria:

- ✓ Functional: The dashboard displays all clusters registered to the tenant with their health status (healthy, warning, critical, offline).
- ✓ Functional: Health status updates in real-time or near real-time (within 30 seconds).
- ✓ Technical: Cluster health data is fetched from the Cluster Management Service via API.
- ✓ Technical: Uses WebSocket or polling mechanism to update cluster health without page reload.
- ✓ Technical: Data is stored and indexed in PostgreSQL with appropriate caching via Redis.

Story Points: 5

Estimated Effort: 5 hours