**ADR-001: Kubernetes with Helm for Fictional College Website Deployment**

**Date:** 2025-05-29  
**Status:** Accepted  
**Deciders:** Development Team

**Context and Problem Statement**

We need to deploy a website for a fictional college that will serve informational content including dorms, study areas, facility hours, move-in tips, and WiFi information. The application requires a reliable, scalable deployment strategy that can handle varying traffic loads and provide easy management of configurations across different environments (development, staging, production).

Key requirements:

* Reliable web application hosting
* Environment-specific configuration management
* Scalability for potential traffic spikes (enrollment periods, move-in days)
* Easy deployment and rollback capabilities
* Infrastructure as Code approach

**Decision Drivers**

* **Scalability**: Need to handle seasonal traffic variations
* **Maintainability**: Easy configuration management across environments
* **Reliability**: High availability for critical student information
* **Developer Experience**: Streamlined deployment process
* **Cost Efficiency**: Reasonable resource utilization
* **Future Growth**: Ability to add new features and services
* **Knowledge Retention**: Mitigate risks of high turnover from college-age student employees who graduate in 4-5 years

**Considered Options**

**Option 1: Traditional Virtual Machine Deployment**

* **Description**: Deploy application on VMs with load balancer
* **Pros**: Simple, familiar technology stack
* **Cons**: Manual scaling, configuration drift, limited automation

**Option 2: Container Orchestration with Docker Compose**

* **Description**: Use Docker Compose for multi-container deployment
* **Pros**: Simple container orchestration, good for development
* **Cons**: Limited production scalability, no built-in service discovery

**Option 3: Kubernetes with kubectl and YAML manifests**

* **Description**: Deploy to Kubernetes using raw YAML configurations
* **Pros**: Full Kubernetes capabilities, explicit configuration
* **Cons**: Configuration duplication, manual template management

**Option 4: Kubernetes with Helm Charts (Selected)**

* **Description**: Use Helm for templated Kubernetes deployments
* **Pros**: Template reuse, environment-specific values, package management
* **Cons**: Additional complexity, Helm-specific learning curve

**Decision**

**We will use Kubernetes with Helm charts for deployment.**

**Reasoning**

1. **Configuration Management**: Helm's templating system allows us to maintain a single chart with environment-specific values files, reducing duplication and configuration drift.
2. **Scalability**: Kubernetes provides horizontal pod autoscaling and load balancing capabilities essential for handling traffic variations during peak periods (enrollment, move-in).
3. **Deployment Management**: Helm's release management provides easy rollbacks, upgrades, and deployment history tracking.
4. **Ecosystem Integration**: Strong ecosystem for monitoring, logging, and security tools that integrate well with Kubernetes.
5. **Future Flexibility**: Can easily add additional services (databases, APIs, microservices) as the college website grows in complexity.
6. **Knowledge Transfer and Documentation**: Helm charts serve as living documentation of deployment processes, reducing the impact of student employee turnover and graduation cycles.

**Consequences**

**Positive**

* **Standardized Deployments**: Consistent deployment process across all environments
* **Easy Scaling**: Automatic horizontal scaling based on traffic patterns
* **Configuration Flexibility**: Environment-specific configurations without code changes
* **Rollback Capability**: Quick rollback to previous versions if issues arise
* **Infrastructure as Code**: Version-controlled deployment configurations
* **Monitoring Integration**: Built-in health checks and metrics collection
* **Knowledge Continuity**: Standardized, documented deployment process reduces dependency on individual team members and eases onboarding of new student employees

**Negative**

* **Complexity**: Higher learning curve for team members unfamiliar with Kubernetes/Helm
* **Resource Overhead**: Kubernetes control plane requires additional infrastructure resources
* **Debugging Complexity**: More layers to troubleshoot when issues occur
* **Helm Dependency**: Additional tool dependency and potential version compatibility issues
* **Initial Setup Time**: Longer initial setup compared to simpler deployment options
* **Staff Turnover Risk**: Complex system may be challenging for incoming student employees to learn quickly, potentially creating knowledge gaps during graduation cycles

**Implementation Plan**

1. **Phase 1**: Set up basic Kubernetes cluster and Helm chart structure
2. **Phase 2**: Implement core website deployment with basic configurations
3. **Phase 3**: Add monitoring, logging, and alerting
4. **Phase 4**: Implement CI/CD pipeline integration
5. **Phase 5**: Add autoscaling and advanced networking features

**Related Decisions**

* **ADR-002** (Future): Container image strategy and registry selection
* **ADR-003** (Future): Monitoring and observability stack selection
* **ADR-004** (Future): CI/CD pipeline implementation approach
* **ADR-005** (Future): SSL/TLS certificate management strategy

**Trade-offs Accepted**

* **Operational Complexity vs. Scalability**: Accepting higher operational complexity in exchange for better scalability and automation
* **Learning Curve vs. Long-term Benefits**: Initial time investment in learning Kubernetes/Helm for long-term deployment benefits
* **Resource Overhead vs. Feature Set**: Additional infrastructure costs for comprehensive container orchestration capabilities
* **System Complexity vs. Staff Continuity**: Accepting higher technical complexity in exchange for standardized processes that survive employee graduation cycles

**Monitoring and Review**

This decision will be reviewed in 6 months or when:

* Team size significantly changes
* Application requirements fundamentally shift
* Major issues arise with the current approach
* New deployment technologies become significantly more attractive