Kubernetes Antipatterns

Kubernetes'te, anti-paternleri tanımlamak ve bunlardan kaçınmak, sağlam bir konteyner orkestrasyon ortamını sürdürmek için çok önemlidir. Bu yanıltıcı uygulamalar başlangıçta etkili görünebilir, ancak komplikasyonlara yol açabilir. Bu okuma, on yaygın Kubernetes anti-paternini keşfeder ve daha sorunsuz ve sürdürülebilir bir dağıtım için alternatif uygulamalar önerir.

1. Konteyner görüntülerinde yapılandırma eklemekten kaçının

Konteynerler, üretim süreci boyunca tutarlı bir görüntü kullanma avantajı sunar. Farklı ortamlar arasında uyum sağlamak için, yapılandırmayı doğrudan konteynerlere gömmeden görüntüler oluşturmak esastır.

Sorun: Görüntüler, test edilen versiyondan sapma gösteren ortam spesifik kalıntılar içerdiğinde sorunlar ortaya çıkar; bu da görüntülerin yeniden inşa edilmesini gerektirir ve üretimde yeterince test edilmemiş versiyonlar riski taşır. Ortam bağımlı konteyner görüntülerinin tanımlanması, sabit IP adresleri, şifreler ve ortam spesifik ön ekler gibi özelliklerin tespit edilmesiyle mümkündür.

En iyi uygulama: Belirli çalışma zamanı ayarlarından bağımsız genel görüntüler oluşturun. Konteynerler, yazılım yaşam döngüsü boyunca tek bir görüntünün tutarlı bir şekilde kullanılmasını sağlayarak basitlik ve verimlilik teşvik eder.

2. Uygulama ve altyapı dağıtımını ayırma

Kod olarak Altyapı (IaC), altyapıyı kod yazarken olduğu gibi tanımlayıp dağıtmayı sağlar. Altyapıyı bir boru hattı aracılığıyla dağıtmak avantajlı olsa da, altyapı ve uygulama dağıtımını ayırmak çok önemlidir.

Sorun: Altyapı ve uygulama dağıtımı için tek bir boru hattı kullanmak, özellikle uygulama kodundaki değişikliklerin altyapı değişikliklerini geride bırakması durumunda kaynak ve zaman israfına yol açar.

En iyi uygulama: Altyapı ve uygulama dağıtımını ayrı boru hatlarına ayırarak verimliliği ve kaynak kullanımını optimize edin.

3. Dağıtımda belirli sıralamanın ortadan kaldırılması

Bağımlılıklardaki gecikmelere rağmen uygulama istikrarını korumak, konteyner orkestrasyonunda kritik öneme sahiptir. Geleneksel sabit başlangıç sıralarının aksine, Kubernetes ve konteynerler bileşenleri aynı anda başlatır.

Sorun: Kötü ağ gecikmesi iletişimi kesintiye uğrattığında zorluklar ortaya çıkar, bu da pod çökmesine veya geçici hizmet kesintisine neden olabilir.

En iyi uygulama: Başarısızlıkları proaktif bir şekilde tahmin edin, kesinti süresini en aza indirmek için çerçeveler oluşturun ve uygulama dayanıklılığını artırmak için bileşenlerin eş zamanlı başlatılması stratejilerini benimseyin.

4. Podlar için bellek ve CPU sınırlarını ayarlama problemi

Belirtilmiş kaynak sınırları olmadan varsayılan Kubernetes ayarı, bir uygulamanın tüm kümeyi potansiyel olarak tekelleştirmesine izin vererek kesintilere neden olabilir.

En iyi uygulama: Tüm uygulamalar için kaynak sınırları belirleyin, her uygulamanın çeşitli koşullar altındaki davranışını kapsamlı bir şekilde inceleyin ve küme performansını optimize etmek için doğru dengeyi sağlayın.

5. Üretim ortamında en son etiketi çekmekten kaçının

Üretim ortamlarında "en son" etiketinin kullanımı, genellikle belirsiz görüntü çekimleri nedeniyle beklenmedik pod çökmesine yol açar. Bu net sürümleme eksikliği, özellikle kesinti çözümü sırasında sorunların hızlı bir şekilde tanımlanmasının kritik olduğu durumlarda sorun gidermeyi zorlaştırır.

En iyi uygulama: Belirli ve anlamlı görüntü etiketleri kullanın; mümkünse derleme tarihini ve saatini de ekleyin. Ayrıca, konteyner görüntülerinin değişmezliğini korumak ve verileri kalıcı depolamada harici olarak saklamak da önemlidir. Dağıtım sonrası konteynerlerde yapılan değişikliklerden kaçınmak, daha güvenli ve tekrarlanabilir dağıtım süreçleri sağlar.

6. Üretim ve üretim dışı iş yüklerini ayırma sorunu

Tüm operasyonel ihtiyaçlar için tek bir kümeye güvenmek zorluklar yaratır. Varsayılan izinlerden kaynaklanan güvenlik endişeleri ve adlandırılmamış Kubernetes kaynaklarıyla ilgili karmaşıklıklar ortaya çıkar.

En iyi uygulama: Sadece üretim amaçları için ikinci bir küme oluşturun, çoklu kiracılıkla ilgili karmaşıklıklardan kaçının. En az iki küme bulundurun—bir üretim için ve bir de üretim dışı için.

7. Kubectl edit/patch problemi ile plansız dağıtımlardan kaçının

Konfigürasyon kayması, birden fazla ortamın plansız dağıtımlar veya değişiklikler nedeniyle farklılaşması durumunda meydana gelir ve bu da dağıtım hatalarına yol açar.

En iyi uygulama: Tüm dağıtımları Git commitleri aracılığıyla gerçekleştirin; böylece kapsamlı bir geçmiş, küme içeriklerinin kesin bilgisi ve ortamların kolayca yeniden oluşturulması veya geri alınması sağlanır.

8. Canlılık ve Hazırlık Propları ile Sağlık Kontrolleri Uygulama Problemi

Sağlık kontrollerinin ihmal edilmesi çeşitli sorunlara yol açabilir. Öngörülemeyen zamanlamalara sahip aşırı karmaşık sağlık kontrolleri, küme içinde iç hizmet reddi saldırılarına neden olabilir.

En iyi uygulama: Her bir konteyner için sağlık probu yapılandırın, canlılık ve hazırlık probu kullanın ve güvenilir uygulama yanıtı için sağlam sağlık kontrollerine öncelik verin.

9. Gizli bilgilerin yönetimini önceliklendirin ve vault sorununu kullanın

Gizli bilgileri doğrudan konteynerlere yerleştirmek kötü bir uygulamadır. Birden fazla gizli bilgi yönetim yöntemi veya karmaşık enjekte etme mekanizmaları, yerel geliştirme ve test süreçlerini karmaşıklaştırabilir.

En iyi uygulama: Tutarlı bir gizli bilgi yönetim stratejisi kullanın, HashiCorp Vault'u düşünün, gizli bilgileri ortamlar arasında tutarlı bir şekilde yönetin ve bunları çalışma zamanında konteynerlere ileterek dayanıklılığı ve güvenliği artırın.

10. Kontrolcüleri kullanın ve her konteyner başına birden fazla işlem çalıştırma sorunundan kaçının

Üretimde doğrudan pod kullanmak sınırlamalar getirir. Pod'lar dayanıklılık, otomatik yeniden planlama ve veri saklama garantileri sunmaz. Kontrolcü olmadan tek bir konteynerde birden fazla işlem çalıştırmak sorunlara yol açabilir.

En iyi uygulama: Bir çoğaltma faktörü ile Deployment kullanın, her konteyner için bir işlem tanımlayın, gerekiyorsa pod başına birden fazla konteyner kullanın ve güvenilirlik ve ölçeklenebilirlik için Deployment, Job veya StatefulSet gibi iş yükü kaynaklarından yararlanın.

Sonuç

Bu Kubernetes anti-paternlerini anlamak ve bunlardan kaçınmak, daha dayanıklı ve verimli bir konteyner orkestrasyon ortamına katkıda bulunur. En iyi uygulamaları benimsemek, daha sorunsuz dağıtımlar sağlar, teknik borcu azaltır ve Kubernetes tabanlı uygulamaların genel istikrarını artırır.

Yazar: Namrah Arif