CKA

Certified Kubernetes Administrator (CKA), Cloud Native Computing Foundation (CNCF) ve The Linux Foundation tarafından sunulan, Kubernetes kümesini yönetme ve sorun giderme konularında uzmanlığınızı kanıtlayan uluslararası geçerliliği olan bir sertifikadır.

CKA'yı diğer sertifikalardan ayıran en önemli özelliği, tamamen uygulamalı (hands-on) bir sınav olmasıdır. Yani, sadece teorik bilgiye değil, aynı zamanda pratik yetkinliğinize, komut satırını hızlı ve etkili kullanma becerinize odaklanır. Sınavda size gerçek bir Kubernetes kümesi ortamı sunulur ve belirli görevleri yerine getirmeniz istenir

CKA

KURUMSAL ÖZEL EĞİTİM

Bu konuyu istediğiniz gibi özelleştirebilirsiniz. Kullandığınız teknolojiye özel uygulamalar ile anlatılmasını, projenize uygun içerik haline getirilmesini ...

Vebende

CKA EĞİTİMİ

Neden CKA?

- **Sektör Talebi:** Kubernetes, modern uygulamaların dağıtımı ve yönetimi için endüstri standardı haline gelmiştir. Bu da Kubernetes bilen ve yönetebilen profesyonellere olan talebi artırmaktadır.
- **Doğrulanmış Yetkinlik:** CKA sertifikası, işverenlere ve meslektaşlarınıza Kubernetes yönetimi konusunda kanıtlanmış becerilere sahip olduğunuzu gösterir.
- **Kariyer Gelişimi:** DevOps Mühendisi, Sistem Yöneticisi, Bulut Mühendisi gibi pozisyonlarda kariyer fırsatlarınızı artırır ve daha yüksek maaş potansiyeli sunar.
- **Pratik Bilgi:** Sınav hazırlık süreci, size Kubernetes'in derinlemesine pratik bilgilerini kazandırır.

KENDİ KENDİNE PRATİK

- **kubectl Hakimiyeti:** kubectl komutlarını çok iyi öğrenmeli ve hızlı kullanabilmelisiniz. Imperative komutlar (kubectl run, kubectl create, --dry-run=client -o yaml vb.) sınavda size çok zaman kazandıracaktır.
- **YAML Okuma/Yazma:** Kubernetes nesnelerini YAML formatında oluşturma, düzenleme ve anlama konusunda yetkin olun.
- **Resmi Kubernetes Dokümantasyonu:** Sınav sırasında sadece kubernetes.io/docs adresini kullanmanıza izin verilir. Bu yüzden, dokümantasyonu hızlıca tarayıp doğru bilgiyi bulma konusunda pratik yapmalısınız. Faydalı sayfaları tarayıcınızda yer imlerine eklemek size büyük avantaj sağlayacaktır.
- **Pratik Ortamlar:** Kendi bilgisayarınızda Minikube veya KIND kurarak veya Play with Kubernetes gibi online laboratuvar ortamlarını kullanarak pratik yapın.
- **Sınav Simülatörleri:** Özellikle Killer.sh, CKA sınavının birebir simülasyonunu sunar. Bu simülatörler, hem sınav ortamına alışmanızı hem de zaman yönetimi becerilerinizi geliştirmenizi sağlar.

EĞİTİM İÇERİĞİ

Modül 1: Kubernetes Dünyasına Giriş ve Temel Mimari

Kubernetes Neden Önemli? Modern Altyapıdaki Yeri

- Kapsayıcıların yönetimi ihtiyacı, mikroservisler ve bulut yerel uygulamaların yükselişi.
- Kubernetes'in sağladığı otomasyon, ölçeklenebilirlik ve dayanıklılık avantajları.

Kubernetes Mimarisi ve Temel Bileşenleri

- Control Plane (Master) ve Worker Node (Minion) kavramları.
- API Server, etcd, Scheduler, Controller Manager bileşenlerinin görevleri ve Control Plane içindeki etkileşimleri.
- Kubelet ve Kube-Proxy bileşenlerinin görevleri ve Worker Node üzerindeki işlevleri.
- *(Hands-on)* Basit bir Kubernetes kümesi ortamına erişim ve temel bileşenlerin durumunu kontrol etme.

Modül 2: Node (İşçi Düğümü) ve Pod (En Küçük Birim) Kavramları

Node (İşçi Düğümü) Kavramı ve Yönetimi

- Pod'ların çalıştığı fiziksel veya sanal sunucular olarak Node'lar.
- Node durumları (Ready, NotReady, MemoryPressure, vb.) ve temel yönetimi.
- Node üzerindeki Kubelet ve Container Runtime (Docker, containerd, CRI-O) ilişkisi.

Pod Yaşam Döngüsü ve Yönetimi

- Kubernetes'in dağıtılabilen en küçük birimi olan Pod'lar ve içerdiği kapsayıcılar.
- Pod durumları (Pending, Running, Succeeded, Failed, Unknown) ve geçişleri.
- Çoklu kapsayıcı Pod'lar: Init Container ve Sidecar pattern'ları.

(Hands-on) Basit Pod oluşturma, durumunu kontrol etme ve silme işlemleri.

Modül 3: kubectl Komut Satırı Araçları ile Etkileşim

kubectl ile Kubernetes Kümesi Yönetimi

- kubectl kurulumu ve yapılandırması (~/.kube/config).
- Kaynakları listeleme (get), detaylarını inceleme (describe).
- Kaynak oluşturma (create), silme (delete).
- Pod içine bağlanma (exec), logları görüntüleme (logs).
- Küme ve Node bilgilerini sorgulama (cluster-info, top nodes).
- *(Hands-on)* Çeşitli kubectl komutları ile temel kaynakları sorgulama ve yönetme pratikleri.

Modül 4: YAML Manifest Dosyaları ile Bildirimsel Yönetim

YAML Formatı ve Kubernetes Manifest Yapısı

- Bildirimsel (declarative) yönetim yaklaşımının önemi.
- YAML dosyasının temel yapısı: apiVersion, kind, metadata, spec, status.
- Ortak metadata alanları: name, namespace, labels, annotations.
- Temel kaynaklar (Pod, Deployment, Service) için YAML örnekleri.

kubectl apply ile Bildirimsel Dağıtım ve Güncelleme

- kubectl apply -f komutu ile YAML dosyalarını uygulama.
- Kaynaklarda yapılan değişiklikleri YAML üzerinden yönetme.
- *(Hands-on)* Basit Pod, Namespace ve ConfigMap gibi kaynakları YAML dosyaları oluşturarak apply komutu ile dağıtma.

Modül 5: Deployment Kaynakları ile Durum Bilgisi Olmayan Uygulama Dağıtımı

Deployment Objelerinin Kullanımı

- Durum bilgisi olmayan (stateless) uygulamaların yönetimi için Deployment'lar.
- İstenen Pod sayısını (replicas) belirleme ve yönetme.
- Deployment'ların ReplicaSet objeleriyle ilişkisi.

Temel Güncelleme Senaryoları

- Yeni sürüm imajı ile Deployment güncelleme.
- Basit geri alma (rollback) işlemleri.
- (Hands-on) Bir web uygulamasını Deployment olarak dağıtma, ölçeklendirme ve temel güncelleme/geri alma senaryolarını uygulama.

Modül 6: Service Kaynakları ve Uygulamalara Erişimin Sağlanması

Service Objelerinin Önemi ve Çalışma Mantığı

- Pod'lara kararlı bir IP adresi ve DNS adı sağlama.
- Service ve Pod'lar arasındaki bağlantı: Selector mekanizması.
- Kube-Proxy'nin Service Discovery ve Load Balancing'deki rolü.

Temel Service Tipleri

- ClusterIP: Küme içinden erişim.
- NodePort: Node IP'si ve belirli bir port üzerinden dışarıdan erişim.
- *(Hands-on)* Dağıtılan uygulamaya ClusterIP ve NodePort Service'leri ile erişim sağlama pratikleri.

Modül 7: Namespace, Label, Selector ve Annotation Kullanımı

Namespace Yapısı ve Kaynak İzolasyonu

- Kaynakları mantıksal gruplara ayırma ve izolasyon sağlama.
- Farklı ortamlarda (dev, staging, prod) Namespace kullanımı.
- kubectl komutlarında Namespace belirtme (-n, --all-namespaces).

Etiketler (Labels) ve Seçiciler (Selectors)

- Objeleri düzenlemek ve gruplamak için Label'lar.
- Service, Deployment gibi kaynakların hedef Pod'ları seçmek için Selector'lar.

Anotasyonlar (Annotations)

- Objeler hakkında ek, tanımlayıcı meta veri ekleme.
- Otomasyon araçları veya operasyonel bilgiler için kullanımı.
- *(Hands-on)* Namespace oluşturma, kaynakları Namespace'ler arasına dağıtma, Label ve Selector kullanarak kaynakları gruplama/seçme.

EĞİTİM İÇERİĞİ

Modül 8: ConfigMap ve Secret ile Konfigürasyon ve Hassas Veri Yönetimi

ConfigMap ile Uygulama Konfigürasyonu Yönetimi
- Konfigürasyon verilerini Pod imajından ayırma.
- Çevre değişkeni veya volume olarak Pod'lara bind etme.

Secret ile Hassas Veri Yönetimi
- Şifreler, API anahtarları gibi hassas bilgileri güvenli saklama.
- Base64 kodlamanın güvenlik sağlamadığının anlaşılması.
- Secret'ları çevre değişkeni veya volume olarak Pod'lara bind etme.
- *(Hands-on)* ConfigMap ve Secret oluşturma, bunları bir uygulamada çevre değişkeni veya dosya olarak kullanma senaryoları.

Modül 9: StatefulSet ve DaemonSet ile Özel Uygulama Tipleri

StatefulSet ile Durum Bilgisi Olan Uygulama Yönetimi
- Veritabanları, mesaj kuyrukları gibi stateful uygulamalar için tasarlanmış objeler.
- Stabil ağ kimlikleri (hostname) ve sıralı dağıtım/ölçekleme garantileri.
- VolumeClaimTemplate ile her Pod için kalıcı depolama atama.

DaemonSet ile Node Başına Tek Pod Dağıtımı
- Her veya belirli Node'larda tek bir Pod çalıştırması gereken arka plan servisleri (log toplayıcılar, izleme ajanları vb.).
- Yeni Node'lar eklendikçe otomatik Pod dağıtımı.
- *(Hands-on)* Basit bir StatefulSet ve DaemonSet dağıtımı, farklılıklarını gözlemleme.

Modül 10: Kalıcı Depolama Yönetimi I: PV ve PVC

Kalıcı Depolama (Persistent Storage) İhtiyacı
- Pod yaşam döngüsünden bağımsız veri saklama gereksinimi.

PersistentVolume (PV) Kavramı
- Kubernetes kümesindeki fiziksel/sanal depolama birimlerini soyutlama.
- PV tipleri (NFS, iSCSI, Cloud Provider Diskler vb.) ve yaşam döngüsü (Available, Bound, Released, Failed).

PersistentVolumeClaim (PVC) Kavramı
- Uygulama (Pod) tarafından talep edilen depolama birimi.
- PV'ler ile PVC'ler arasındaki binding (eşleşme) süreci.
- *(Hands-on)* Statik PV oluşturma, bu PV'ye bir PVC ile bağlanma ve bir Pod'dan bu PVC'yi kullanma.

Modül 11: Kalıcı Depolama Yönetimi II: StorageClass ve Dinamik Tedarik

StorageClass Kaynakları
- Farklı depolama tiplerini (SSD, HDD, performans seviyeleri) tanımlama şablonları.
- Tedarikçi (Provisioner) kavramı ve rolü.

Dinamik Tedarik (Dynamic Provisioning)
- PVC oluşturulduğunda StorageClass kullanarak PV'nin otomatik olarak oluşturulması.
- Varsayılan StorageClass atama.

Depolama Alanı Büyütme (Volume Expansion)
- PVC boyutunu artırma yetenekleri (eğer StorageClass destekliyorsa).
- *(Hands-on)* StorageClass tanımı (basit bir provizyoner ile), Dinamik Tedarik kullanarak PVC oluşturma ve bir Pod'dan bu PVC'yi kullanma.

Modül 12: Ingress Kaynakları ve Harici Erişimi Yönetme

Ingress Kavramı ve Hizmeti
- HTTP/HTTPS trafiğini küme içindeki Service'lere yönlendirme.
- Service'lerin NodePort veya LoadBalancer tiplerine alternatif esnek bir çözüm.
- Ingress Controller'lar (Nginx Ingress, Traefik, HAProxy vb.) ve görevleri.

Ingress Kaynağı Tanımlama
- Hostname ve Path tabanlı yönlendirme kuralları.
- TLS (HTTPS) sonlandırma yapılandırması.
- Birden çok Service'e tek bir Ingress üzerinden erişim sağlama.
- *(Hands-on)* Bir Ingress Controller kurulumu (varsa), Ingress kaynakları tanımlayarak farklı URL'ler üzerinden küme içindeki uygulamalara erişim sağlama.

Modül 13: Kaynak Yönetimi (Requests, Limits) ve Sağlık Kontrolleri (Probes)

Kaynak İstekleri (Requests) ve Limitleri (Limits)
- Pod'ların ihtiyaç duyduğu minimum (Requests) ve kullanabileceği maksimum (Limits) CPU ve bellek (memory) kaynaklarının tanımlanması.
- Planlama (Scheduling) ve kaynak tahsisindeki etkileri.
- Overcommitment ve Quality of Service (QoS) sınıfları.

Uygulama Sağlık Kontrolleri: Liveness ve Readiness Probları
- Liveness Probe: Kapsayıcının çalışıp çalışmadığını kontrol etme, başarısız olursa kapsayıcıyı yeniden başlatma.
- Readiness Probe: Kapsayıcının trafiği kabul etmeye hazır olup olmadığını kontrol etme, hazır değilse Service'ten düşürme.
- Farklı Probe tipleri (HTTPGet, TCPSocket, Exec).
- *(Hands-on)* Kaynak istekleri/limitleri tanımlanmış Pod'lar dağıtma, Liveness ve Readiness Probları yapılandırarak uygulama sağlığını yönetme senaryoları.

EĞİTİM İÇERİĞİ

Modül 14: Planlama Kısıtlamaları (Taints, Tolerations) ve İlişkilendirme (Affinity)

Taints ve Tolerations
- Node'lara "leke" (taint) ekleyerek Pod'ların o Node'da çalışmasını engelleme.
- Pod'lara "tolerans" (toleration) ekleyerek belirli taint'lere sahip Node'larda çalışmasına izin verme.
- Özel Node havuzları oluşturma (GPU Node'lar, özel donanımlar).

Node ve Pod İlişkilendirmesi (Affinity/Anti-Affinity)
- Pod'ları belirli Node'larda (Node Affinity) veya belirli Pod'ların yakınında/uzağında (Pod Affinity/Anti-Affinity) çalışmaya zorlama veya tercih etme.
- Uygulama dayanıklılığını artırma (Pod Anti-Affinity ile farklı Node'lara dağıtma).
- *(Hands-on)* Node'lara taint ekleme, Pod'lara toleration ve affinity kuralları uygulayarak planlama davranışını gözlemleme.

Modül 15: Rol Tabanlı Erişim Kontrolü (RBAC) Temelleri ve Servis Hesapları

Kubernetes Güvenliği: Kimlik Doğrulama ve Yetkilendirme
- API Server'a erişim güvenliği mekanizmaları (TLS, Sertifikalar).
- Kimlik Doğrulama (Authentication): Kim olduğunu kanıtlama (Kullanıcılar, Servis Hesapları).
- Yetkilendirme (Authorization): Ne yapabileceğine karar verme (RBAC).

Rol Tabanlı Erişim Kontrolü (RBAC) Objeleri
- **Role ve ClusterRole: Kaynaklar üzerindeki izinleri tanımlama.**
- RoleBinding ve ClusterRoleBinding: Tanımlanan izinleri Kullanıcılara veya Servis Hesaplarına atama.

Servis Hesapları (ServiceAccount)
- Uygulamaların (Pod'ların) API Server ile etkileşim kurmak için kullandığı kimlikler.
- Pod'lara Servis Hesabı atama.
- *(Hands-on)* Temel RBAC Role, RoleBinding tanımları yaparak belirli kullanıcılara veya servis hesaplarına Namespace veya küme genelinde kaynaklara erişim izinleri verme.

Modül 16: Pod Güvenliği: Security Context ve Secret Yönetimi Best Practices

Security Context Kavramı
- Pod veya Kapsayıcı düzeyinde uygulanan güvenlik ayarları.
- runAsUser, runAsGroup, fsGroup ile kullanıcı/grup id'leri.
- Linux Yetenekleri (Capabilities).
- İmtiyazlı (Privileged) kapsayıcılar.

Secret Yönetimi Güvenlik Best Practice leri
- Secret'ların oluşturulması ve dağıtılmasına yönelik güvenli yaklaşımlar.
- Secret verilerinin Etcd'de şifrelenmesi (Encryption at Rest) kavramı ve önemi.
- *(Hands-on)* Security Context ayarları ile Pod dağıtımı, temel Secret yönetimi güvenlik adımları (Etcd şifrelemesi kavramsal).

Modül 17: Kubernetes Ağ Mekanizmaları: CNI, DNS ve Ağ Politikaları

CNI (Container Network Interface) ve Ağ Modelleri
- Kubernetes'in farklı ağ eklentileriyle (Calico, Flannel, Cilium vb.) nasıl etkileşim kurduğu.
- Pod'lar arası ve Pod-dış dünya arası ağ trafiğinin temel akışı ve farklı CNI modelleri (Overlay, BGP).

Küme İçi DNS Mekanizması (CoreDNS)
- CoreDNS'in çalışması ve DNS kayıtlarının (Service, Pod) çözümlenmesi.
- Uygulamaların diğer Servisleri veya Pod'ları isimle bulması.

Ağ Politikaları (NetworkPolicy) ile İletişim Güvenliği
- Pod'lar arasındaki ağ iletişimini (ingress/egress) kısıtlama ve segmentasyon.
- Güvenlik duvarı kuralları gibi NetworkPolicy tanımlama.
- *(Hands-on)* Küme içi DNS sorgulama, basit NetworkPolicy tanımları yaparak Pod'lar arası iletişimi kısıtlama senaryoları.

Modül 18: Küme Operasyonları: etcd Yönetimi, Yükseltme ve Sorun Giderme

etcd Yönetimi
- Kubernetes kümesinin ana veri deposu etcd'nin önemi.
- etcd'nin yedeklenmesi ve geri yüklenmesi stratejileri ve araçları (etcdctl).
- etcd kümesi sağlık kontrolleri.

Küme Bakımı ve Yükseltmeler (Overview)
- kubeadm veya bulut sağlayıcısı araçları ile küme yükseltme adımlarına genel bakış.
- Yükseltme öncesi ve sonrası dikkat edilmesi gerekenler.

İleri Düzey Küme ve Bileşen Sorun Giderme
- Control Plane bileşenleri (API Server, Scheduler, Controller Manager, etcd) sorunlarını teşhis etme.
- Node kaynak (CPU, bellek, disk) sorunlarını giderme.
- Karmaşık ağ ve depolama sorunlarının teşhisi.
- *(Hands-on)* etcd yedekleme/geri yükleme senaryosu (eğitim ortamında simülasyon), yaygın küme bileşeni hatalarını giderme pratikleri.

EĞİTİM SÜRESİ

- 3 Gün
- Ders Süresi: 50 dakika
- Eğitim Saati: 10:00 - 17:00

Eğitim formatında eğitimler 50 dakika + 10 dakika moladır. 12:00-13:00 saatleri arasında 1 saat yemek arasındaki verilir. Günde toplam 6 saat eğitim verilir. 3 günlük formatta 18 saat eğitim verilmektedir.

Eğitimler uzaktan eğitim formatında tasarlanmıştır. Her eğitim için teams linkleri gönderilir. Katılımcılar bu linklere girerek eğitimlere katılırlar. Ayrıca farklı remote çalışma araçları da eğitmen tarafından tüm katılımlara sunulur. Katılımcılar bu araçları kullanarak eğitimlere katılırlar.

Eğitim içeriğinde github ve codespace kullanılır. Katılımcılar bu platformlar üzerinden örnek projeler oluşturur ve eğitmenle birlikte eğitimlerde sorulan sorulara ve taleplere uygun iceriğe cevap verir. Katılımcılar bu araçlarla eğitimlerde sorulan sorulara ve taleplere uygun iceriğe cevap verir.
Eğitim yapay zeka destekli kendi kendine öğrenme formasyonu ile tasarlanmıştır. Katılımcılar eğitim boyunca kendi kendine öğrenme formasyonu ile eğitimlere katılırlar. Bu eğitim formatı sayesinde tüm katılımcılar gelecek tüm yaşamlarında kendilerini güncellemeye devam edebilecekler ve her türlü sorunun karşısında çözüm bulabilecekleri yeteneklere sahip olacaklardır.

EĞİTİM YÖNTEMİ

- **Temel Linux & Komut Satırı:** Katılımcıların Linux komutlarına ve terminal kullanımına hakim olması. Vim/Nano gibi editörleri etkin kullanabilmesi.
- **Konteyner & Docker:** Konteyner kavramları ve Docker temel bilgisi.
- **Temel Ağ Bilgisi:** IP, port, DNS gibi ağ kavramlarına aşinalık.
- **Temel YAML Bilgisi:** YAML dosyalarını okuma ve düzenleme becerisi.
- **Problem Çözme:** Analitik düşünme ve sorun giderme yeteneği.
- **Pratik İsteği:** Sürekli öğrenmeye açık olma ve yoğun pratik yapma motivasyonu.

KATILIMCILARDAN BEKLENTİLERİMİZ

1. Temel Linux Bilgisi ve Komut Satırı Yetkinliği
Kubernetes'in Linux üzerinde çalışması sebebiyle, katılımcıların temel Linux komutlarına (ls, cd, cp, mv, rm, mkdir vb.) ve komut satırı (terminal) kullanımına hakim olmaları kritik önem taşır. Sınav tamamen terminal üzerinde ilerleyeceği için, Vim veya Nano gibi metin editörlerini etkin kullanabilmek de büyük bir avantaj sağlayacaktır.

2. Konteyner Temelleri ve Docker Bilgisi
Kubernetes'in konteyner orkestrasyonu aracı olması nedeniyle, katılımcıların Docker gibi bir konteyner platformunun temel kavramlarını (imajlar, konteynerler) anlamaları ve temel Docker komutlarını (docker run, docker ps) çalıştırabilmeleri beklenir. Bu bilgi, Kubernetes'teki Pod'lar ve Deployment'lar gibi yapıları kavramayı kolaylaştıracaktır.

3. Temel YAML Bilgisi
Kubernetes kaynak tanımları genellikle YAML formatında yazılır. Katılımcıların YAML'ın temel yapısını (girinti, anahtar-değer çiftleri, listeler) okuyup anlayabilmeleri ve basit YAML dosyalarını oluşturup düzenleyebilmeleri önemlidir.

4. Problem Çözme ve Analitik Düşünme Becerisi
CKA sınavının önemli bir kısmı sorun giderme (Troubleshooting) üzerine kuruludur. Katılımcıların karşılaştıkları problemleri analiz etme, mantıksal adımlar izleyerek kök nedeni bulma ve etkili çözümler üretme becerisine sahip olmaları beklenir. Resmi Kubernetes dokümantasyonunu (kubernetes.io/docs) etkin kullanabilme de bu süreçte hayati rol oynar.

5. Yoğun Pratik Motivasyonu
CKA sertifikasyonunda başarı, sadece dersleri takip etmekle değil, yoğun ve düzenli pratikle gelir. Katılımcılarımızın eğitimde verilen tüm uygulamalı laboratuvar çalışmalarını eksiksiz yapmaları, hatta kendi başlarına (Minikube, KIND gibi araçlarla) veya sınav simülatörleri (örn. Killer.sh) üzerinde bolca pratik yapmaya istekli olmaları en büyük beklentimizdir. Sınavın zaman kısıtlaması nedeniyle, hızlı ve doğru işlem yapma alışkanlığı kazanmak kritik öneme sahiptir.

HEDEF KİTLE

1. DevOps Mühendisleri
Uygulama dağıtımı, otomasyonu ve sürekli entegrasyon/sürekli teslimat (CI/CD) süreçlerinden sorumlu olan DevOps mühendisleri için CKA, Kubernetes ortamlarını daha etkin yönetme ve optimize etme yetkinliği kazandırır.

2. Sistem Yöneticileri (SysAdmins)
Sunucuların, ağ altyapısının ve uygulama platformlarının kurulumu, bakımı ve sorun gidermesinden sorumlu sistem yöneticileri, CKA ile konteynerize edilmiş iş yüklerini ve Kubernetes kümelerini yönetme becerilerini geliştirirler.

3. Bulut Mühendisleri
Genel bulut platformlarında (AWS, Azure, GCP) çalışan ve bulut tabanlı altyapıları tasarlayan, uygulayan ve yöneten mühendisler için CKA, bulut yerel (cloud-native) uygulamaları ve Kubernetes servislerini daha iyi yönetmelerini sağlar.

4. Yazılım Geliştiriciler (DevOps Rolüne Geçmek İsteyenler)
Konteyner ve Kubernetes dünyasına ilgi duyan, geliştirdikleri uygulamaları Kubernetes üzerinde dağıtmak ve yönetmek isteyen yazılım geliştiriciler için CKA, operasyonel süreçleri anlama ve DevOps rollerine geçiş yapma konusunda güçlü bir temel sunar.

5. IT Mimarları ve Çözüm Mimarları
Kurumsal IT altyapısını veya bulut çözümlerini tasarlayan mimarlar için CKA, Kubernetes'in mimari yapısını, güvenlik ve ölçeklenebilirlik prensiplerini derinlemesine anlayarak daha sağlam ve optimize edilmiş çözümler sunmalarına yardımcı olur.

6. Kubernetes Öğrenmek İsteyen Herkes
Yukarıdaki unvanlara sahip olmasa da, güçlü bir Linux temeli ve öğrenme motivasyonu olan herkes CKA eğitimine katılabilir. Özellikle kariyerini bulut bilişim ve modern altyapı yönetimi alanında şekillendirmek isteyen profesyoneller için CKA, sektörde aranan bir yetkinlik kapısıdır.

"

Kurumsal size özel eğitimler hazırlıyoruz. Her eğitim yeni bir heyecan.

Vebende A.Ş.

Kurumsal Terzi Usulü Butik Eğitimler.

Size özel hazırlanan seminer, danışmanlık, eğitim ve hizmetlerimizle yüksek verim elde edin. Paranızı boşa harcamayın. Zaman çok değerli.

Her Eğitim Yeni Bir Heyecan

2000 yılından günümüze devam eden eğitim heyecanı. Uluslararası tecrübe, proje geliştirme deneyimleri, danışmanlıklar ve arge mühendislik deneyimlerimizi ülkemize sunmak için yeni bir konsept tasarladık.
Sizi dinliyor, takımınıza uygun özel içerikler ile hazırlanmış eğitimler hazırlıyoruz. Her eğitim özel bir çalışma, içerik üretimi, uygulama örnekleri hazırlıkları, sunumlar hazırlamayı gerektiriyor. Aynı eğitimi yönetim kadrosuna farklı, teknik ve mimar ekiplerinize farklı içerikler ile hazırlıyoruz. Her eğitim yeni bir macera ve heyecan.

Bizimle İletişime Geçin

iletisim@vebende.com.tr

www.vebende.com.tr

İzmir - Türkiye

Kurumsal
Eğitimler

www.vebende.com.tr

İletişime Geçin

Whatsapp: 0542 5505704
Tel: 0532 512 7811

iletisim@vebende.com.tr

www.vebende.com.tr

Adalet Mah. Manas Bulvarı
No:39 İç Kapı N0: 3107 Bayraklı
İZMİR

[Sürekli Güncellenen Eğitimlerimizi Sitemizden Takip Edin.]

"

Kurumsal terzi usulu size özel eğitimler ve uluslararası deneyime sahip eğitmenler ile çalışmanın keyfi

"

Size Neler Sunuyoruz?

- Kitaptan okunan eğitimler sunmuyoruz. Sizin için özel hazırlanan içerikler ve uygulamalı eğitimler hazırlıyoruz.
- Her eğitim için sunumlar, materyaller, örnek senaryolar size özel hazırlıyoruz.
- Her eğitim için katılımcılara özel github repoları hazırlanıyor. Eğitimden sonra da katılımcılar hayat boyu kaynaklara ulaşmaya devam edebilsinler diye, **"katılımcılar ve eğitmenle birlikte yapılan tüm çalışmaları"** github repolarında saklıyoruz.

Misyonumuz

1. Ülkemizde dünyanın en güncel teknolojilerini kazandırmak. En güncel teknolojilerine hakim takımlarına katkı sağlamak.
2. Alışılmışın dışında en güncel teknolojileri deneyimli eğitmenler ve uygulamalar ile sunmak.
3. Siber güvenlik konusunda hassas çalışmalar sunarak, çağımızın büyük kabusu siber tehditler konusunda deneyimli bilgili takımların oluşmasına katkı sağlamak.